Configuring Collectors

This chapter describes the format of the XML files which are used to set up and collect data from the collectors. The XML files describe the collectors by specifying schedules, devices, data handlers, notifiers and collector names. Each collector type has its own set of parameters that must be defined.

Several of the collectors support Hotspot Polling, Data Exporting, Data Purging, and Threshold Processing. Cisco CNS PerfE also has built-in VoIP templates that you can use to set up collectors. Refer to Chapter 5, "Features Supported by Collectors," for specific information on the XML format for these features.

This chapter is divided into sections, one for each type of collector. Each section describes that collector's unique attributes and how to set them.

BAMS Collector

BAMS is the Billing and Measurement Server. BAMS is an adjunct application for the PGW 2200 and it converts Cisco proprietary Call Detail Record (CDR) input files from the PGW 2200 format into industry standard or custom output formats. BAMS also produces trunk group-level operational measurements. The BAMS Collector collects call records in ASCII format from BAMS and uses FTP to transfer the files.

Requirements

The schedule specified for a BAMS collector should be the same or a multiple of the collection period the BAMS device is configured with. One FTP-type device definition must be referenced. If the customer setup includes a backup BAMS, the <alternateBAMS> tag in <bamsCollector> should be specified.

Only one instance of the BAMS Collector should be configured per Cisco CNS PerfE system. The BAMS Collector should be used only in conjunction with the RADIUS Collector.

For detailed information on configuring BAMS, see the document *Billing and Measurement Server User's Guide*.

If the customer has a backup BAMS, the retrieval schedule, file prefix, file suffix, and storage directories must be configured to be the same on both devices.

BAMS must be configured to clean up the ASCII files after some period of time (e.g., scheduled every one or two days). The BAMS Collector will not mark files for cleanup by renaming them. Note that if other customer equipment also reads these files, that equipment must not rename or modify these files.

XML Configuration

The following tags can be configured for a BAMS Collector:

Table 6-1 Tags in the XML Configuration for the BAMS Collector

XML Tag	Purpose	Mandatory/ Optional
alternateBAMS	Host name or IP address of the alternate BAMS, if one is available.	Optional
asciiFilePrefix	File prefix used for BAMS ASCII output files, typically *cdr_*.	Mandatory
asciiFileSuffix	File suffix used for BAMS ASCII output files, typically *.csv*.	Mandatory
asciiFileDirectory	Directory on the BAMS device where ASCII call record files are stored, typically **/opt/CiscoBAMS/data/ASCII.**.	Mandatory

The following is a sample XML configuration for the BAMS Collector (schedule and device must be defined prior to collector):

```
<collector name="bams1">
      <bamsCollector>
            <schedule name="Schedule15Minutes"/>
            <device name="dev1"/>
            <alternateBAMS>192.168.1.100</alternateBAMS>
            <asciiFilePrefix>cdr_</asciiFilePrefix>
            <asciiFileSuffix>.csv</asciiFileSuffix>
            <asciiFileDirectory>/opt/CiscoBAMS/data/ASCII </asciiFileDirectory>
      </bamsCollector>
</collector>
```

Data Export, Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The BAMS Collector does not persist any data retrieved from a BAMS device. As soon as the data is retrieved, the correlation module in the RADIUS Collector correlates this data with RADIUS and Call History data. The BAMS Collector does not support any data export, purging, hotspot polling or threshold processing.

Limitations

None.

BTS Collectors

This section assumes one is familiar with the BTS 10200 product. Please refer to the following web link for more information on BTS 10200 product:

http://www.cisco.com/univercd/cc/td/doc/product/voice/index.htm

Note A CCO username and password is required to access the BTS 10200 documentation.

BtsBilling Collector

The BtsBilling Collector waits for CNS PerfE or the Billing Mediator to FTP billing records over to the Cisco CNS PerfE and processes the records according to the XML configuration.

The BtsQosProcessor plugin will extract only QoS Information from the records and export this data. CNS PerfE does not configure BTS EMS.

Requirements

Note This requirement is needed if BTS EMS will be sending CDRs directly to the CNS PerfE host. If a mediator is placed between BTS EMS and CNS PerfE, refer to the mediator's User Guide to determine how to configure the BTS EMS host.

The Billing Account Address must be configured at the BTS EMS host manually. Use the **change billing-acct-addr** command to send a billing record to the Cisco CNS PerfE host:

```
change billing-acct-addr;
billing-file-prefix=<cnsperfe->;
billing-server-addr=<CNS_PerfE_host-ip-address>;
billing-server-directory=<CNS_PerfE_Home>/data/bts/cdr;
user-name=dasadmin;
password=<dasadmin-password>;
polling-internal=15;
```

XML configuration

```
<das>
      <create>
            <schedule name="15M">
                  <interval>PT15M</interval>
            </schedule>

            <schedule name="30M">
                  <interval>PT30M</interval>
            </schedule>

            <device name="BtsHost">
                  <telnet>
                        <ipaddress>10.0.0.1</ipaddress>
                        <username>user1</username>
                        <password>pass1</password>
                  </telnet>
                  <ftp>
                  </ftp>
            </device>

            <dataHandler name="BtsBillingHandler">
                  <url>
                        <prefix>ftp://user2:pass2@10.0.0.4/%2Fdata/</prefix>
                  </url>
            </dataHandler>

            <processor name="BtsQos">
                  <plugin name="QOS">
                        <class>com.cisco.das.BTSCollector.BtsQosProcessor</class>
                        <parameter name="version">3.3</parameter>
                  </plugin>
            </processor>

            <collector name="BtsBilling">
                  <BtsBillingCollector>
                        <processor name="BtsQos"/>
                        <schedule name="15M"/>
                        <device name="BtsHost"/>
                        <dataHandler name="BtsBillingHandler">
                              <schedule name="30M"/>
                        </dataHandler>
                  </BtsBillingCollector>
             </collector>

      </create>
      <add/>
      <start>
            <collector name="BtsBilling"/>
      </start>
```

```
</das>
```

Data Export

The BtsQosProcessor plugin only exports the following information:

```
<HEADER>COLLECT_TIMESTAMP,BILLING_CALL_TYPE,BILLING_CALL_ELAPSED_T
IME,BILLING_ORIG_NUMBER,BILLING_TERM_NUMBER,ORIG_QOS_TIME,TERM_QOS
_TIME,ORIG_QOS_PACKETS_SENT,ORIG_QOS_PACKETS_RECD,ORIG_QOS_OCTETS_S
ENT,ORIG_QOS_OCTETS_RECD,ORIG_QOS_PACKETS_LOST,ORIG_QOS_JITTER,ORIG_
QOS_AVG_LATENCY,TERM_QOS_PACKETS_SENT,TERM_QOS_PACKETS_RECD,TERM
_QOS_OCTETS_SENT,TERM_QOS_OCTETS_RECD,TERM_QOS_PACKETS_LOST,TERM_
QOS_JITTER,TERM_QOS_AVG_LATENCY</HEADER>
<DATA>2002-10-19 19:31:19.88,3,00:01:20,9723701417,,2,,1,2,0,0,0,0,66,0,0,0,0,0,
0,0</DATA>
```

If the BtsBilling Collector is configured without the BtsQosProcessor plugin then the collector will export the billing records in the same format as the BTS EMS does. Refer to the BTS 10200 documentation for complete format of the billing records.

Threshold Crossing Alerts

The BtsBilling Collector supports thresholds in Qos data only.

```
<threshold name=”QoSOrigJitter”>
   <attribute name=ORIG_QOS_JITTER”>
         <raiseOperation>GTE</raiseOperation>
         <raiseValue>500</raiseValue>
         <clearOperation>LT</clearOperation>
         <clearValue>200<</clearValue>
         <level>major</level>
  </attribute>
</threshold>
```

BtsBilling Collector Plugin

The BtsBilling Collector supports one Processor plugin called BtsQosProcessor.

When this processor plugin is configured the BtsBilling Collector will only export QoS data, otherwise it will export the complete billing records received from the BTS EMS system.

Properties file

das.properties:

```
bts.billing.ftp.dir=<billing records ftp over from bts ems or mediation system>
bts.billing.file.prefix=<billing file prefix>
```

Note These values should match the values specified in the **change billing-acct-addr** command only if CDRs are sent directly to the BTS EMS system.

BtsTm Collector

The BtsTm Collector retrieves various Traffic Measurement (TM) reports from the BTS EMS host. The following twelve traffic measurement categories are available from the BTS 10200 EMS:

Measurement Data Type	Measure Data Description
ISDN	ISDN signaling protocol-related information
CALLP	Call Processing specific information
MGA	MGCP signaling protocol-related information
SIM	Service Interaction Manager related information
POT-FS	POTS/Centrex/Tandem Feature related information
AIN-FS	AIN Feature Server related information
ISUP	ISDN User Part (SS7) signaling protocol information
TG-USAGE	Trunk Group usage-related information
ANM	Announcement Server related information
H323	H.323 signaling protocol-related information
BILLING	Call Detail Data related information
SIA	Signaling Interface Adapter related information

For an overview of the various measurement data reports for the BTS 10200, see Appendix C, “Measurement Data Report Formats”.

XML configuration

```
<das>
      <create>
            <schedule name="S1">
                  <start>2003-01-01T00:05:00.000-00:00</start>
                  <interval>PT15M</interval>
            </schedule>

            <schedule name="S2">
                  <start>2003-01-01T00:07:00.000-00:00</start>
                  <interval>PT15M</interval>
            </schedule>

            <device name="BtsHost">
                  <telnet>
                        <ipaddress>10.0.0.1</ipaddress>
                        <username>optiuser</username>
                        <password>optiuser</password>
                        <prompt>CLI</prompt>
                  </telnet>
                  <ftp>
                        <ipaddress>10.0.0.1</ipaddress>
                        <username>optiuser</username>
                        <password>optiuser</password>
```

```
            </ftp>
        </device>

        <dataHandler name="BtsTmDH">
            <ftp>
                <urlPrefix>ftp://user1:pass1@10.0.0.1/data/</urlPrefix>
            </ftp>
        </dataHandler>

        <notifier name="BtsNotifier">
            <cns>
                <subject>cns.cisco.das.listener</subject>
            </cns>
        </notifier>

        <threshold name="MGA_TPM_UNREACHABLE">
            <attribute name="mga.MGA_TPM_UNREACHABLE">
                <raiseOperation>GT</raiseOperation>
                <raiseValue>0</raiseValue>
                <clearOperation>EQ</clearOperation>
                <clearValue>0</clearValue>
                <level>minor</level>
            </attribute>
        </threshold>

        <collector name="BtsTm01">
            <BtsTmCollector>
                <schedule name="S1"/>
                <device name="BtsHost"/>
                <threshold name="MGA_TPM_UNREACHABLE"/>
                <notifier name="BtsNotifier"/>
                <dataHandler name="BtsTmDH">
                    <schedule name="S2"/>
                </dataHandler>
                <tmType>mga</tmType>
                <tmType>tg-usage</tmType>
                <standbyEMS>10.0.0.2</standbyEMS>
            </BtsTmCollector>
        </collector>
    </create>

    <start>
        <collector name="BtsTm01"/>
    </start>
</das>
```

The BTS-EMS generates reports every 15 minutes. Therefore, the scheduler "S1" is configured to collect data from the BTS-EMS every 15 minutes plus some offset. The offset in this case is 5 minutes as indicated by the <start> tag.

Data Export

The file contains the name of the report, in this case MGA, then attr names are specified in the header tag followed by the actual data values specified in the data tag.

```
<MGA>
<HEADER>MGA_TIMESTAMP,MGA_TPM_DECODER_ERROR,MGA_TPM_ENC
ODE_ERROR,MGA_TPM_UNREACHABLE,MGA_TPM_SEND_FAILED,MGA_TP
M_CRCX_ACK_RECVD,MGA_TPM_CRCX_NACK_RECVD,MGA_TPM_CRCX_S
ENT,MGA_TPM_MDCX_ACK_RECVD,MGA_TPM_MDCX_NACK_RECVD,MGA
_TPM_MDCX_SENT,MGA_TPM_DLCX_RECVD,MGA_TPM_DLCX_SENT,MGA_
TPM_DLCX_ACK_RECVD,MGA_TPM_RQNT_ACK_RECVD,MGA_TPM_RQNT_
```

```
SENT,MGA_TPM_AUEP_ACK_RECVD,MGA_TPM_AUEP_NACK_RECVD,MGA_
TPM_AUEP_SENT,MGA_TPM_NTFY_RECVD,MGA_TPM_RSIP_RECVD,MGA_T
PM_RSIP_ACK_SENT</HEADER>
<DATA>2002-05-05 16:15:00,0,0,0,0,30,0,30,50,0,50,0,32,32,0,189,0,189,0,0,0
</DATA>
</MGA>
```

Each exported data file will have the following format:

```
<reportName>
      <header>col1Name, col2Name,….colXName</header>
      <data>col1Data,col2Data,….colXData</data>
</reportName>
```

The number of columns within the <data> tag might be less than the number of columns in the <header> tag depending on the BTS 10200 version used.

Threshold Crossing Alerts and Purging Data

The collector supports thresholds. The threshold attribute name format is as follows:

```
<report-name>.<attr-name>
```

Example:

```
mga.MGA_TPM_UNREACHABLE
```

See the XML configuration for a complete example (above).

The following is the syntax for thresholds relating to TG-USAGE reports:

```
tg-usage.<attrName>.<grptype>.<grpid>
```

Thresholds for all trunk groups:

```
tg-usage.<attrName>
```

Thresholds for a particular trunk group:

```
tg-usage.<attrName>.<grptype>
```

Threshold for a particular trunk group type and id:

```
tg.usage.<attrName>.grptype>.<grpid>
```

The BtsTm Collector supports purging of collected performance data. This is done on a schedule configured for all of Cisco CNS PerfE or based on an individual purger.

Properties file

das.properties:

```
bts.tm.report.dir=<bts ems dir where reports are located>
bts.tm.file.prefix=<report file prefix>
bts.cnspe.tmp.dir=<where to ftp the reports from bts ems>
```

BulkMIB Collector

Many Cisco devices support SNMP for performance data retrieval. Retrieving large tables via SNMP results in many SNMP **getbulk** requests and has more overhead. So, many Cisco devices support bulk file transfer to retrieve large tables. The device has to be configured to indicate the objects to retrieve via the bulk transfer facility, and when the transfer is requested the device writes these objects to the specified file and transfers the file via FTP to the requested host. The BulkMIB Collector limits bulk file transfer to only tables (the OIDs specified in the BulkMIB Collector tag must be the OIDs of a table).

The BulkMIB Collector uses **CISCO-BULK-MIB.my** and **CISCO-FTP-CLIENT.my** MIBs at the device. During each collection period, the BulkMIB Collector configures the bulk MIB and FTP client MIB, requests that the file be created, and requests that the file is sent using FTP. It then removes the configuration at the device. So, there is certain overhead associated for Cisco CNS PerfE with setting up bulk MIB for each collection and the device has to generate and FTP the file for each collection. Due to these reasons, it is recommended that bulk MIB transfer be used for retrieving large tables with reasonable collection frequency.

In the BulkMIB Collector, the FTP-related parameters ipaddress, username, and password are made configurable through XML. These are optional parameters.

If these parameters are not provided in the configuration, then the values from **das.properties** will be used (**bulkmib.ftp.username, bulkmib.ftp.password**). These parameters are maintained for backward compatibility and are deprecated in 2.0.

Except for data collection, the BulkMIB Collector and MIB Collector behave similarly with respect to threshold processing, data export, data purge, etc.

Requirements

Set the limits for the tables in these MIBs appropriately (**cbfDefineMaxFile**, **cbfDefineMaxObjects**, **cbfStatusMaxFiles**, **cfcRequestMaximum**).

The properties **bulkmib.ftp.username** and **bulkmib.ftp.password** are stored in **CISCO-FTP-CLIENT-MIB.my**.

The user account used for FTP should have write privileges to the ***<CNS_PerfE_Home>*/tmp** directory. Furthermore, the FTP daemon should be enabled on the Cisco CNS PerfE host.

In addition, take note of the following:

- Cisco IOS software release 12.0 or later is recommended. Earlier IOS images might fail with the BulkMIB Collector.
- Check the MIB Locator tools to determine if the CISCO-BULK-FILE-MIB and CISCO-FTP-CLIENT-MIB MIBs are supported:

  http://www.cisco.com/warp/public/477/nms_tools.shtml
- Bulk MIB is not supported on Catalyst OS devices.
- SNMP must be configured on the device with the read-write community string set. Check the following web location on how to configure SNMP:

  http://www.cisco.com/warp/public/477/SNMP/12.html
- Router does not support SFTP. FTP is used to transfer data from the device to the CNS PerfE host.

Note that in CNS PerfE, the **dasadmin** account is used by BulkMib Collector to ftp data to CNS PerfE host. If CNS-PerfE is installed under **/opt**, ftp fails, because **dasadmin** cannot access the **/opt** directory. As a result, the **/etc/ftpaccess** file should have the **dasadmin** user in the access list as follows.

```
realuser root dasadmin
```

This enables **dasadmin** user to access root directories, viz. /opt.

XML Configuration

The following XML tags can be configured for a BulkMIB Collector:

Table 6-2 Tags in the XML Configuration for the Bulk MIB Collector

XML Tag	Purpose	Mandatory/ Optional
localFtpServer ipaddress username password	The name or IP address of the local FTP server. Name of the host where the Cisco CNS PerfE software is running in order for the network device to FTP the file over. The username and password correspond to the user account to FTP files to the Cisco CNS PerfE host. When not specified, the username and password configured in **das.properties** is used.	Optional
oid	The OIDs specified in the BulkMIB Collector tag must be the OID of a table.	Mandatory.

Sample XML configuration:

```
<collector name="BulkMibCollect">
      <BulkMibCollector>
      <schedule name="BulkMibSchedule"/>
      <device name="BulkMibDevice"/>
      <dataHandler name="BulkMibExport"/>
      <localFtpServer>
            <ipaddress>10.77.12.94</ipaddress>
            <username>dasadmin</username>
            <password>dasadmin</password>
      </localFtpServer>
      <oid>atTable</oid>
      </BulkMibCollector>
</collector>
```

Data Export, Threshold Crossing Alerts, Purging Data, and Hotspot Polling

Refer to the MIB Collector sections: "Data Export" section on page 6-57 and "Threshold Crossing Alerts, Purging Data, and Hotspot Polling" section on page 6-58.

Limitations

Cisco CNS PerfE does not support retrieval of SNMP OPAQUE data types. Cisco CNS PerfE supports only table transfers in bulk transfer file.

BulkMIB Collector Plugins

MIB Collector Plugins are also supported in the BulkMIB Collector configuration. The difference is that the BulkMIB Collector does not support querying scalar MIB attributes. For example the **sysUpTime** attribute which is used in computation in the MIB Collector cannot be queried using the BulkMIB Collector. As a result, the **collectionTime** attribute is used for computations. This is the time interval between two successive polls.

Since collectionTime attribute is in seconds when compared to sysUpTime which is in hundredths of a second, computation can use collectionTime value directly. Hence there is slight change in the processor components with regard to this.

Templates for different processor components for BulkMIB Collector are as follows:

1. Delta Plugin

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifInOctets</parameter>
                        <parameter name="delta.2">collectionTime</parameter>

                        <parameter name="result.1">deltaIfInOctets delta.1</parameter>
                        <parameter name="result.2">deltaCollectionTime delta.2</parameter>

                  </plugin>
            </processor>
      </create>
</das>
```

2. Percentage In Link Utilization

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifInOctets</parameter>
                        <parameter name="delta.2">collectionTime</parameter>
                        <parameter name="var.1">ifSpeed</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">8</parameter>
                        <parameter name="op.1">divide delta.1 delta.2</parameter>
                        <parameter name="op.2">multiply op.1 int.2</parameter>
                        <parameter name="op.3">multiply op.2 int.1</parameter>
                        <parameter name="op.4">divide op.3 var.1</parameter>
                        <parameter name="result.1">pctInLinkUtilization op.4</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

3. Percentage Out Link Utilization

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifOutOctets</parameter>
                        <parameter name="delta.2">collectionTime</parameter>
                        <parameter name="var.1">ifSpeed</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">8</parameter>
                        <parameter name="op.1">divide delta.1 delta.2</parameter>
                        <parameter name="op.2">multiply op.1 int.2</parameter>
                        <parameter name="op.3">multiply op.2 int.1</parameter>
                        <parameter name="op.4">divide op.3 var.1</parameter>
                        <parameter name="result.1">pctOutLinkUtilization op.4</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

4. Percentage In Link Utilization with Configured Bandwidth

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifInOctets</parameter>
                        <parameter name="delta.2">collectionTime</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">512000</parameter>
                        <parameter name="op.1">multiply delta.1 int.1</parameter>
                        <parameter name="op.2">multiply delta.2 int.2</parameter>
                        <parameter name="op.3">divide op.1 op.2</parameter>
                        <parameter name="result.1">pctInLinkUtilBW op.3</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

5. Percentage Out Link Utilization with configured bandwidth

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifOutOctets</parameter>
                        <parameter name="delta.2">collectionTime</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">512000</parameter>
                        <parameter name="op.1">multiply delta.1 int.1</parameter>
                        <parameter name="op.2">multiply delta.2 int.2</parameter>
                        <parameter name="op.3">divide op.1 op.2</parameter>
                        <parameter name="result.1">pctOutLinkUtilBW op.3</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

6. AAA Server plugin

```
<das>
      <load>
            <mib name="CISCO-AAA-SERVER-MIB.my"/>
      </load>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">casAuthenTransactionFailures</parameter>
                        <parameter
name="delta.2">casAuthenTransactionSsuccesses</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="op.1">multiply delta.1 int.1</parameter>
                        <parameter name="op.2">plus delta.1 delta.2</parameter>
                        <parameter name="op.3">divide op.1 op.2</parameter>
                        <parameter name="result.1">pctCasAuthenFailuresBW op.3</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

Sample configuration:

```
<das>
      <create>
            <schedule name="BulkMibSchedule">
                  <start>2002-03-06T00:00:00.000-08:00</start>
                  <interval>PT2M</interval>
                  <interTaskDelay>PT0.1S</interTaskDelay>
            </schedule>

            <device name="BulkMibDevice">
                  <snmp>
                        <ipaddress>10.77.11.70</ipaddress>
                        <readCommunity>public</readCommunity>
                        <writeCommunity>private</writeCommunity>
                  </snmp>
            </device>

            <dataHandler name="BulkMibExport">
                  <url>
                        <prefix>ftp://mramacha:cisco123@10.77.11.94/%2Fdata/MibPlugin</pre
fix>
                  </url>
            </dataHandler>

            <collector name="BulkMibCollect">
                  <BulkMibCollector>
                        <processor name="P1"/>
                        <schedule name="BulkMibSchedule"/>
                        <device name="BulkMibDevice"/>
                        <dataHandler name="BulkMibExport"/>
                        <oid>ifTable</oid>
                  </BulkMibCollector>
            </collector>
      </create>
      <start>
            <collector name="BulkMibCollect"/>
      </start>
</das>
```

ByteCap Data Collector

The ByteCap Data Collector provides information about the amount of data traffic to and from a cable modem. This information can be used to identify can be used to identify usage characteristics of a subscriber along with potential abusers of the terms of service, e.g. upstream data transfers identify a large upstream traffic transfer which might indicate the subscriber is running servers.

The ByteCap Data Collector collects this information from the CMTS using SNMP and the data can be transferred via framework hotspot and periodic polling.

Requirements

The ByteCap Data Collector requires the framework definition of the CMTSs as a device. See XML configuration below.

XML Configuration

The following is a sample XML configuration for the BDC (schedule, device and data handlers must be defined prior to collector). The database must contain an entry for uBR01.

Define the device to the framework as an SNMPv3 capable device:

```
<device name="ubr1">
  <snmp>
    <ipaddress>10.5.1.2</ipaddress>
    <readCommunity>public1</readCommunity>
    <timeout>PT5S</timeout>
    <retry>4</retry>
    <version>SNMPv3</version>
    <port>161</port>
    <userName>HORSHAM</userName>
    <authProtocol>MD5</authProtocol>
    <authPassword>HORSHAM1</authPassword>
  </snmp>
</device>
```

There are algorithms in the ByteCap collector that allow the higher level system to identify a three tiered summaries of data:

- Daily
- Weekly
- Monthly

These are purely ascii mnemonics for ease of use. The algorithms defined are:

- MonthlySummary= SUM(numberOfPeriods) = # of weekly summaries in a month
- WeeklySummary= SUM(numberOfUnits) = # of daily summaries in a week
- DailySummary= SUM(interval) = # hours in a day

As an example:

```
Bytecap collector is set to run every 4 hours
Set Daily "numberOfCollections" = 6 ( * 4 hours = 24hours)
Set Weekly "numberOfUnits" = 7 ( * 24 hours = 7 days)
Set Monthly "numberOfPeriods" = 4 ( * 7 days = 28 days)
```

For each interval expiration, the data will exported to a specified location.

Raw data collected is automatically purged when **numberOfCollections** has expired and only the last **numberOfCollections** worth of raw data is maintained.

Daily summaries are automatically purged and only the last **numberOfUnits** are stored. The purge method of Weekly summaries is automatic after Monthly has fired. The purge method of Monthly summaries are under control of the framework purge.

Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<das>
    <create>
     <notifier name="CNSnotifier_bytecap">
        <cns>
            <subject>cns.cisco.das.listener_bytecap</subject>
        </cns>
      </notifier>
      <dataHandler name="hotspot_bytecap">
        <cns>
          <subject>cns.cisco.das.listener_bytecap</subject>
        </cns>
      </dataHandler>
     <purger name="purge_bytecap">
         <time>2002-10-23T15:00:00.000-05:00</time>
         <interval>PT1440H</interval>
         <delay>PT720H</delay>
      </purger>
      <collector name="bytecap1">
        <DocsisByteCapCollector>
         <schedule name="sch4hr"/>
         <device name="ubr1"/>
         <device name="ubr2"/>
        <notifier name="CNSnotifier_bytecap"/>
         <dataHandler name="hotspot_bytecap"/>
         <purger name="purge_bytecap"/>
         <byteCapSummary>
            <dataHandler name="hotspot_bytecap"/>
            <numberOfPeriods>4</numberOfPeriods>
            <period>
               <dataHandler name="hotspot_bytecap"/>
               <numberOfUnits>7</numberOfUnits>
               <unit>
                  <dataHandler name="hotspot_bytecap"/>
                  <numberOfCollections>6</numberOfCollections>
               </unit>
            </period>
         </byteCapSummary>
        </DocsisByteCapCollector>
      </collector>
      </create>
    <add/>
    <start>
       <collector name="bytecap1"/>
        </start>
</das>
```

Hotspot and Periodic Exported Data

```
<BYTECAP>
cmtsip|macaddress|lastcollectedtime|totalinoctets|totaloutoctets
10.87.117.2|00:02:16:d5:a4:0d|2003-03-21 09:01:54.356|2119769|1315060
```

```
10.87.117.2|00:04:27:a5:c0:45|2003-03-21 09:01:54.326|2242028|1385945
10.87.117.2|00:04:27:a5:c0:81|2003-03-21 09:01:54.34|2189981|1314685
10.87.117.2|08:00:3e:08:ef:4c|2003-03-21 09:01:54.376|1897436|1436540
10.87.117.2|00:08:0e:16:fb:be|2003-03-21 09:01:54.363|2689849|0
10.87.117.2|00:04:bd:b7:81:ca|2003-03-21 09:01:54.393|2689908|0
10.87.117.2|00:02:16:d5:a4:27|2003-03-21 09:01:54.323|2367606|1314639
</BYTECAP>
```

Summary Exported Data

The following are snippets of the data available via Hotspot or Periodic based upon the defined summarization:

Daily:

```
<BYTECAP title="daily">
cmtsip|macaddress|lastcollectedtime|totalinoctets|totaloutoctets
10.87.117.2|00:04:27:a5:c0:81|2003-03-21 09:01:54.34|2189981|1314685
10.87.117.2|00:02:16:d5:a4:27|2003-03-21 09:01:54.323|2367606|1314639
10.87.117.2|08:00:3e:08:ef:4c|2003-03-21 09:01:54.376|1897436|1436540
10.87.117.2|00:02:16:d5:a4:0d|2003-03-21 09:01:54.356|2119769|1315060
10.87.117.2|00:08:0e:16:fb:be|2003-03-21 09:01:54.363|2689849|0
10.87.117.2|00:04:27:a5:c0:45|2003-03-21 09:01:54.326|2242028|1385945
10.87.117.2|00:04:bd:b7:81:ca|2003-03-21 09:01:54.393|2689908|0
</BYTECAP>
```

Weekly:

```
<BYTECAP title="weekly">
cmtsip|macaddress|lastcollectedtime|totalinoctets|totaloutoctets
10.87.117.2|00:02:16:d5:a4:0d|2003-03-21 09:01:54.356|2119769|1315060
10.87.117.2|00:08:0e:16:fb:be|2003-03-21 09:01:54.363|2689849|0
10.87.117.2|00:04:27:a5:c0:45|2003-03-21 09:01:54.326|2242028|1385945
10.87.117.2|00:04:bd:b7:81:ca|2003-03-21 09:01:54.393|2689908|0
10.87.117.2|00:04:27:a5:c0:81|2003-03-21 09:01:54.34|2189981|1314685
10.87.117.2|00:02:16:d5:a4:27|2003-03-21 09:01:54.323|2367606|1314639
10.87.117.2|08:00:3e:08:ef:4c|2003-03-21 09:01:54.376|1897436|1436540
</BYTECAP>
```

Monthly:

```
<BYTECAP title="monthly">
cmtsip|macaddress|lastcollectedtime|totalinoctets|totaloutoctets
10.87.117.2|00:04:27:a5:c0:45|2003-03-21 09:01:54.326|2242028|1385945
10.87.117.2|08:00:3e:08:ef:4c|2003-03-21 09:01:54.376|1897436|1436540
10.87.117.2|00:04:bd:b7:81:ca|2003-03-21 09:01:54.393|2689908|0
10.87.117.2|00:02:16:d5:a4:0d|2003-03-21 09:01:54.356|2119769|1315060
10.87.117.2|00:04:27:a5:c0:81|2003-03-21 09:01:54.34|2189981|1314685
10.87.117.2|00:08:0e:16:fb:be|2003-03-21 09:01:54.363|2689849|0
10.87.117.2|00:02:16:d5:a4:27|2003-03-21 09:01:54.323|2367606|1314639
</BYTECAP>
```

The interval of data in historical storage for Monthly is available via XML:

Note The **<start>** and **<end>** tags and data are ignored for the collection interval query.

```
<get>
     <data>
       <collector name="bytecap1"/>
       <device name="ubr1"/>
       <start>2002-06-27T12:30:00.000-08:00</start>
```

```
      <end>2003-08-27T13:30:00.000-08:00</end>
      <url>ftp://guest:guest@161.44.33.23/%2F/home/guest/das/data1/bytecap.txt</url>
     </data>
    </get>
```

The stored historical information can be retrieved on demand by higher level applications.

Monthly Data:

```
<das>
    <get>
     <data>
       <collector name="bytecap1"/>
       <device name="ubr1"/>
       <start>2002-06-27T12:30:00.000-08:00</start>
       <end>2003-08-27T13:30:00.000-08:00</end>

<url>ftp://guest:guest@161.44.33.23/%2F/home/guest/das/data1/bytecap-monthly.txt</url>
       <parameter name="docsisQueryType">
            docsis-bytecap-monthly
       </parameter>
     </data>
    </get>
```

Weekly data.

```
    <get>
     <data>
       <collector name="bytecap1"/>
       <device name="ubr1"/>
       <start>2002-06-27T12:30:00.000-08:00</start>
       <end>2003-08-27T13:30:00.000-08:00</end>

<url>ftp://guest:guest@161.44.33.23/%2F/home/guest/das/data1/bytecap-weekly.txt</url>
       <parameter name="docsisQueryType">
            docsis-bytecap-weekly
       </parameter>
     </data>
    </get>
```

Daily data.

```
    <get>
     <data>
       <collector name="bytecap1"/>
       <device name="ubr1"/>
       <start>2002-06-27T12:30:00.000-08:00</start>
       <end>2003-08-27T13:30:00.000-08:00</end>

<url>ftp://guest:guest@161.44.33.23/%2F/home/guest/das/data1/bytecap-daily.txt</url>
       <parameter name="docsisQueryType">
            docsis-bytecap-daily
       </parameter>
     </data>
    </get>
```

CallHistory Collector

The CallHistory Collector retrieves call history information for telephony and VoIP call logs at the originating and terminating voice gateways. The CallHistory Collector accomplishes this task by polling the following Managed Information Bases (MIBs):

- CISCO-DIAL-CONTROL-MIB
- CISCO-VOICE-DIAL-CONTROL-MIB
- CISCO-VOICE-COMMON-DIAL-CONTROL-MIB

Gateways have a configurable buffer size to hold the call history information and time allotted to retain this data. Cisco CNS PerfE should be configured so that Cisco CNS PerfE retrieves the call history information while it is still available at the device.

Note With IOS releases starting from 12.2(11)T, all the QoS parameters that are available in the Call History MIB are also available as Radius Vendor-Specific Attributes. It is, therefore, recommended that the Radius Collector be used instead of the CallHistory Collector to reduce load at the MIB.

The CallHistory Collector retrieves the telephony and VoIP call legs from the Call History tables, correlates these two legs for each call and writes the resulting data to a temporary file in **<*CNS_PerfE_Home*>/data/callhistory** directory.

The CallHistory Collector should only be used in conjunction with the RADIUS Collector so that the results from the CallHistory Collector can be appended and correlated to the data collected through RADIUS CDRs. The third-party application must ensure that only one CallHistory Collector is configured on each Cisco CNS PerfE system.

The CallHistory Collector automatically determines how many entries to poll by retrieving the **maxTableEntries** value from the gateway.

Requirements

The Radius Collector must be configured and running before the CallHistory Collector because the CallHistory Collector relies on the Radius Collector to generate the export data files.

XML Configuration

The following XML example shows typical configuration for a Call History Collector. The following attributes must be specified in the configuration: **connection-id** and **h323-call-origin**. Based on the load of the network, and the impact of polling on the device, other attributes can be omitted. See the Attribute Mapping table in the "CallHistory Collector Attributes" section on page B-1.

```
<collector name="callHistoryCollector">
      <callHistoryCollector>
            <schedule name="POLLER3MIN"/>
            <callHistAttrs>
                  <generic>
                        <attr>h323-setup-time</attr>
                        <attr>peer-address</attr>
                        <attr>peer-subaddress</attr>
                        <attr>peer-id</attr>
                        <attr>peer-if-index</attr>
                        <attr>logical-if-index</attr>
```

```
                        <attr>disconnect-cause</attr>
                        <attr>disconnect-text</attr>
                        <attr>h323-connect-time</attr>
                        <attr>h323-disconnect-time</attr>
                        <attr>h323-call-origin</attr>
                        <attr>charged-units</attr>
                        <attr>info-type</attr>
                        <attr>bytes_out</attr>
                        <attr>paks_out</attr>
                        <attr>bytes_in</attr>
                        <attr>paks_in</attr>
                    </generic>
                    <voip>
                        <attr>connection-id</attr>
                        <attr>h323-remote-address</attr>
                        <attr>remote-udp-port</attr>
                        <attr>round-trip-delay</attr>
                        <attr>selected-qos</attr>
                        <attr>session-protocol</attr>
                        <attr>session-target</attr>
                        <attr>ontime-rv-playout</attr>
                        <attr>gapfill-with-prediction</attr>
                        <attr>gapfill-with-interpolation</attr>
                        <attr>gapfill-with-redundancy</attr>
                        <attr>hiwater-playout-delay</attr>
                        <attr>lowater-playout-delay</attr>
                        <attr>receive-delay</attr>
                        <attr>vad-enable</attr>
                        <attr>coder-type-rate</attr>
                        <attr>h323-voice-quality</attr>
                        <attr>lost-packets</attr>
                        <attr>early-packets</attr>
                        <attr>late-packets</attr>
                    </voip>
                    <telephony>
                    <!-- Telephony Attributes -->
                        <attr>connection-id</attr>
                        <attr>tx-duration</attr>
                        <attr>fax-tx-duration</attr>
                        <attr>noise-level</attr>
                        <attr>acom-level</attr>
                        <attr>session-target</attr>
                        <attr>img-pages-count</attr>
                    </telephony>
                </callHistAttrs>
            </callHistoryCollector>
</collector>
```

Data Export, Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The data retrieved from the CallHistory Collector is meant for Cisco CNS PerfE use. Therefore, the CallHistory Collector does not support data export, threshold processing, or hotspot polling. Call history data will be correlated with RADIUS CDRs as soon as the data is retrieved and hence purging also does not apply to the CallHistory Collector.

Limitations

The CallHistory Collector polls all the devices at the same interval regardless of the type of device. So, the schedule for the CallHistory Collector should be based on the buffer size for call history tables at the device.

CallMgrCdr Collector

The Cisco Call Manager provides voice call information data in two forms. It generates Call Detail Records (CDRs) for each call placed to and from the IP Phones, conferences bridges and off-net PSTN gateways. In addition, it generates the Call Management Record (CMR) associated with each CDR record. The CDRs and corresponding CMR records provide voice quality statistics. The CMR records contain the count of bytes sent, packet sent, jitter and latency, dropped packet etc.

The CMR records are normally generated for each call when a call involves an IP phone as an endpoint. If both endpoints are IP Phones then there will be two CMR records generated, one for each phone.

The Cisco CallMgrCdr Collector provides the following features:

- XML interface to configure the collector from a local/remote host
- Hourly and daily summary reports
- Periodic and On-Demand export of Summary & CDR reports
- Thresholds supported on hourly and daily reports
- Different purge levels for different reports
- Supports multiple Call Manager CDR collectors

Cisco Call Manager Setup Requirements

The following parameters should be enabled on the Cisco Call Manager System in order for the CallMgrCdr Collector to collect the CDRs and CMRs from the Cisco Call Manager's database. Make sure that these parameters are enabled.

- CdrEnabled – enables/disables CDR records.
- CallDiagnosticsEnabled – enables/disables CMR records
- SQL Authentication– enable SQL authentication so that CNS PerfE can CDRs/CMRs via JDBC.
- Microsoft's SQL JDBC Driver

  JDBC driver is not shipped with the product. You need to download the driver from Microsoft's website. Please refer to the Installation Guide for more information on how to download and install the JDBC driver.

CallMgrCdr Collector Setup

The CallMgrCdr Collector retrieves the data from the Cisco Call Manager CDR database. The collector must wait for Call Manager to populate the CDRs into the CDR database after the voice call has been completed. As a result, the interval delay settings for CDR and CMR collection are specified by the following parameters:

CDR collection range is determined as follows:

starttime = currenttime – collectionOffset – collectionScheduleInterval

endtime = currenttime + collectionScheduleInterval

The <collectMode> tag options:

CDR – collect only CDRs

CDRPlusCMR – Collect CDRs Plus CMRs

XML configuration

```
<das>
             <load/>
             <create>
                 <schedule name="S1">
                     <start>2003-01-01T00:00:00.000-08:00</start>
                     <interval>PT15M</interval>
                  </schedule>
                 <device name="CCMDev">
                     <jdbc>
                    <url
name="dbCDR">jdbc:microsoft:sqlserver://10.0.0.1:1433;Databasename=CDR;
          USER=CiscoCCMCDR;PASSWORD=dipsy</url>
                     </jdbc>
                 </device>
                 <dataHandler name="cdrDH">
                     <url>
                         <prefix>file:///export/data/callmgr/cdr_</prefix>
                     </url>
                 </dataHandler>
                 <dataHandler name="hourlySummaryDH">
                     <url>
                         <prefix>file:///export/data/callmgr/hourly_</prefix>
                     </url>
                 </dataHandler>
                 <dataHandler name="dailySummaryDH">
                     <url>
                         <prefix>file:///export/data/callmgr/daily_</prefix>
                     </url>
                 </dataHandler>
                 <notifier name="callmgrNotifier">
                     <cns>
                         <subject>cns.cisco.das.listener</subject>
                     </cns>
                 </notifier>
                 <threshold name="HourlyThresholds">
                     <attribute name="HourlySummary.avgJitter">
                         <raiseOperation>EQ</raiseOperation>
                         <raiseValue>8</raiseValue>
                         <clearOperation>GT</clearOperation>
                         <clearValue>4</clearValue>
                         <level>minor</level>
```

```
                    </attribute>
                </threshold>
                <collector name="CCM01">
                    <CallMgrCdrCollector>
                        <schedule name="S1"/>
                        <device name="CCMDev"/>
                        <threshold name="HourlyThresholds"/>
                        <notifier name="callmgrNotifier"/>
                        <dataHandler name="cdrDH">
                            <parameter name="report">CDRPlusCMR</parameter>
                        </dataHandler>
                        <dataHandler name="hourlySummaryDH">
                            <parameter name="report">HourlySummary</parameter>
                        </dataHandler>
                        <dataHandler name="dailySummaryDH">
                            <parameter name="report">DailySummary</parameter>
                        </dataHandler>
                        <collectMode>CDRPlusCMR</collectMode>
                      <collectionOffset>PT15M</collectionOffset>
                      <hourlyReport>enabled</hourlyReport>
                      <dailyReport>enabled</dailyReport>
                      <cdrReportPurgeLevel>PT24H</cdrReportPurgeLevel>
                      <hourlyReportPurgeLevel>PT168H</hourlyReportPurgeLevel>
                      <dailyReportPurgeLevel>PT336H</dailyReportPurgeLevel>
                    </CallMgrCdrCollector>
                </collector>
            </create>
            <add/>
            <stop/>
            <start>
                <collector name="CCM01"/>
            </start>
        </das>
```

Data Export:

The CallMgrCdr Collector exports CDRs and Summary reports (hourly and daily). The report parameter configured with the data handler controls the data export functionality. Each data handler supports one type of data export report. The parameter tag in the data handler determines what kind of report is to be exported. The same data handler can not be used to export different kinds of reports in a given CallMgrCdr Collector.

The CallMgrCdr Collector supports the following reports:

- CDR: <parameter name=”report”>CDR</parameter>
- CDR: Plus CMR <parameter name=”report”>CDRPlusCMR</parameter>
- Hourly Summary: <parameter name=”report”>HourlySummary</parameter>
- Daily Summary: <parameter name=”report”>DailySummary</parameter>

The following section describes the contents of the various data export reports.

CDR Report Format:

```
<report name=”cdr” version=”1”>
   <collector name=”callmgrCdrCollector”/>
   <device name=”callmgrDev”>
      <call callId=”xxx” callManagerId=”xxx” clusterId=”xxx”>
         <cdr callId=”xxx” callManagerId=”xxx” clusterId=”xxx”>
            <!-- callStatus: completed, abandoned, or failed ->
            <callStatus>completed</callStatus>
```

```
            <!-- gatewayCall: did the call involve a gateway: no, originating, terminating
->
             <gatewayCall>originating</gatewayCall>
             <cdrData>x1,x2,...</cdrData>
          </cdr>
       </call>
   </device>
</report> [padmat1]
```

For the description of the fields specified in the **<cdrData>** tag, refer to the following documentation:

Call Detail Record Definition for Release 3.3(2)

http://www.cisco.com/univercd/cc/td/doc/product/voice/vpdd/cdd/3_3/3_3_2/cdr332.pdf

CDR Plus CMR Report Format:

```
<report name="cdrPlusCmr" version="1">
   <collector name="callmgrCdrCollector"/>
   <device name="callmgrDev">
      <call callId="xxx" callManagerId="xxx" clusterId="xxx">
         <cdr callId="xxx" callManagerId="xxx" clusterId="xxx" >
            <!-- callStatus: completed, abandoned, or failed ->
            <callStatus>completed</callStatus>
            <!-- gatewayCall: did the call involve a gateway: no, originating, terminating
->
            <gatewayCall>originating</gatewayCall>
            <cdrData>x1,x2,...</cdrData>
            <cmr>
               <cmrData>y1,y2,...</cmrData>
               <cmrData>y1,y2,...</cmrData>
            </cmr>
         </cdr>
      </call>
   </device>
</report>
```

For the description of the fields specified in the **<cmrData>** tag, refer to the following documentation:

Call Detail Record Definition for Release 3.3(2)

http://www.cisco.com/univercd/cc/td/doc/product/voice/vpdd/cdd/3_3/3_3_2/cdr332.pdf

Summarized Hourly Report Format:

recordVersion,recordType,collectorName,deviceName,dateTimeStamp,totalAttemptedCalls, totalCompletedCalls,totalAbandonedCalls,totalFailedCalls,totalIncomingGWCalls, totalOutgoingGWCalls, totalGWCalls, maxDuration, minDuration,avgDuration, maxJitter,maxLatency,maxPacketLost, maxPacketSent,maxOctetsSent, maxPacketRecv, maxOctetsRecv,avgJitter,avgLatency,avgPacketLost, avgPacketSent,avgOctetsSent, avgPacketRecv,avgOctetsRecv

Summarized Daily Report Format

version,recordType,collectorName,deviceName,timestamp,totalAttemptedCalls, totalCompletedCalls, totalAbandonedCalls, totalFailedCalls,totalIncomingGWCalls, totalOutgoingGWCalls,totalGWCalls,maxDuration,minDuration,avgDuration,

maxJitter,maxLatency,maxPacketLost,maxPacketSent,maxOctetsSent,maxPacketRecv,
maxOctetsRecv,avgJitter, avgLatency,avgPacketLost,avgPacketSent, avgOctetsSent,
avgPacketRecv,avgOctetsRecv

CDRPlusCMR Report Format

```
<report name="cdrPlusCmr" version=”1”>
    <collector name="CCM01"/>
    <device name="CCMDev">
    <call callId="6" callManagerId="1" clusterId="StandAloneCluster">
    <cdr callId="6" callManagerId="1" clusterId="StandAloneCluster">
      <callStatus>completed</callStatus>
      <gatewayCall>terminating</gatewayCall>

<cdrData>1,6,16777227,1060883906,1,0,1250323328,0,5000,0,16,1250323328,4001,4,20,196274356
0,16777228,1,0,1250323328,0,5001,5001,0,0,1250323328,4002,4,20,1962743560,1060883906,10608
83909,5001,8e885cd8-9c29-436a-8da8-9a098045cab0,,,,,3,SEP000011110001,SEP000011110002,0,0,
0,12,0,0,0,StandAloneCluster,0</cdrData>
      <cmr>

<cmrData>1,6,1,5000,16777227,1060883909,295,49468,277,46531,0,0,0,fa03d143-f96d-4c16-a215-
8c22baf8bc46,,SEP000011110001,StandAloneCluster</cmrData>

<cmrData>1,6,1,5001,16777228,1060883909,295,49468,277,46531,0,0,0,92e01fa1-112c-4b31-95ce-
b388cdb86130,,SEP000011110002,StandAloneCluster</cmrData>
   </cmr>
   </cdr>
   </call>
   <call callId="5" callManagerId="1" clusterId="StandAloneCluster">
   <cdr callId="5" callManagerId="1" clusterId="StandAloneCluster">
      <callStatus>completed</callStatus>
<gatewayCall>terminating</gatewayCall>
<cdrData>1,5,16777225,1060883901,1,0,1250323328,0,5008,0,16,1250323328,4009,4,20,4990698,1
6777226,1,0,1250323328,0,5005,5005,0,0,1250323328,4010,4,20,4990698,1060883902,1060883905,
5005,76468193-e695-494a-aa49-40763a368aca,,,,,3,SEP000011110009,SEP00001111000A,0,0,0,12,0
,0,0,StandAloneCluster,0</cdrData>
      <cmr>

<cmrData>1,5,1,5008,16777225,1060883905,295,49468,277,46531,0,0,0,d6afe4c4-21a8-4507-97a9-
5d71dcf72c1a,,SEP000011110009,StandAloneCluster</cmrData>

<cmrData>1,5,1,5005,16777226,1060883905,295,49468,277,46531,0,0,0,42e206a1-212b-4f9e-aa42-
f04d96de589b,,SEP00001111000A,StandAloneCluster</cmrData>
   </cmr>
   </cdr>
   </call>
</device>
</report>
```

Hourly Report Format

```
recordVersion,recordType,collectorName,collectorDevice,dateTimeStamp,totalAttemptedCalls,t
otalCompletedCalls,totalAbandonedCalls,totalFailedCalls,totalIncomingGWCalls,totalOutgoing
GWCalls,totalGWCalls,maxDuration,minDuration,avgDuration,maxJitter,maxLatency,maxPacketLos
t,maxPacketSent,maxOctetsSent,maxPacketRecv,maxOctetsRecv,avgJitter,avgLatency,avgPacketLo
st,avgPacketSent,avgOctetsSent,avgPacketRecv,avgOctetsRecv
1,3,CCM01,CCMDev,1060880400,10,10,0,0,0,10,10,3,3,3,0,0,0,295,49468,277,46531,0,0,0,295,49
468,277,46531
```

Threshold Crossing Alerts

The Call Manager CDR Collector will support thresholds on summarized data. The format of the threshold name will be the name of the summarized report plus the attribute in that report.

For example:

```
<threshold name="hourlyAverageJitter">
     <attribute name="hourlySummary.avgJitter">

</threshold>

<threshold name="dailyFailedCalls">
     <attribute name="dailySummary.totalFailedCalls">

</threshold>
```

The following thresholds are supported:

- **totalFailedCalls**
- **maxJitter**
- **maxLatency**
- **maxPacketLost**
- **avgJitter**
- **avgLatency**
- **avgPacketLost**

CBQoS Collector

The purpose of the CBQoS collector is to collect ClassMap and Police Statistics from the CLASS BASED QoS (CBQos) MIB implemented on the gateway.

The Modular Qos CLI (MQC) is a CLI structure that allows users to create traffic service policies and attach these service policies to interfaces. A traffic policy contains a traffic class and one or more QoS features. A traffic class is used to classify traffic, while the QoS feature in the traffic class determines how to treat the classified traffic.

Please refer to the following link for more information on MQC:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fqos_c/fqcprt8/

XML Configuration

```
<das>
      <create>
            <schedule name="15min">
                  <interval>PT15M</interval>
            </schedule>

            <schedule name=20min">
                  <interval>PT20M</interval>
            </schedule>

            <device name="cbqos-dev">
```

```
                  <snmp>
                        <ipaddress>10.0.1.71</ipaddress>
                        <readCommunity>public</readCommunity>
                        <writeCommunity>public</writeCommunity>
                  </snmp>
            </device>

            <dataHandler name="CBQoSExporter">
                  <url>
                        <prefix>ftp://user:passwd@host/incoming/cbQoS/</prefix>
                  </url>
            </dataHandler>
            <notifier name="notifier1">
                  <cns>
                        <subject>cns.cisco.das.listener</subject>
                  </cns>
            </notifier>

            <threshold name="CBQoSThresholds">
                  <attribute name="55.DT.queue1.cbQosCMPrePolicyPkt">
                        <raiseOperation>GT</raiseOperation>
                        <raiseValue>300</raiseValue>
                        <clearOperation>LT</clearOperation>
                        <clearValue>4</clearValue>
                        <level>minor</level>
                  </attribute>
            </threshold>

            <collector name="CBQoS">
                  <CBQoSCollector>
                        <schedule name="15min"/>
                        <device name="cbqos-dev"/>
                        <threshold name="CBQoSThresholds"/>
                        <notifier name="notifier1"/>
                        <dataHandler name="CBQoSExporter">
                              <schedule name="20min"/>
                        </dataHandler>
                  </CBQoSCollector>
            </collector>

      </create>
      <add/>
      <start>
            <collector name="CBQoS"/>
      </start>
</das>
```

Thresholds

The collector supports thresholds. The threshold attribute name format is as follows:

```
<attribute name="ifIndex.servicePolicyName.classMapName.statName">
```

Refer to the example above.

The format supports both **servicePolicyName** and **policeStatName**.

- **servicePolicyName** - the name used in the policy-map command in the IOS configuration file.
- **policeStatName** - the name of the police statistics variable.

See the CISCO-CLASS-BASED-QOS-MIB.my MIB for more detailed information.

Supported threshold names:

- cbQosCMPrePolicyPkt
- cbQosCMPrePolicyByte
- cbQosCMPrePolicyBitRate
- cbQosCMPostPolicyByte
- cbQosCMPostPolicyBitRate
- cbQosCMDropPkt
- cbQosCMDropByte
- cbQosCMDropBitRate

- cbQosPoliceConformedPkt
- cbQosPoliceConformedByte
- cbQosPoliceConformedBitRate
- cbQosPoliceExceededPkt
- cbQosPoliceExceededByte
- cbQosPoliceExceededBitRate
- cbQosPoliceViolatedPkt
- cbQosPoliceViolatedByte
- cbQosPoliceViolatedBitRate

- cbQosQueueingCurrentQDepth
- cbQosQueueingMaxQDepth
- cbQosQueueingDiscardByte64
- cbQosQueueingDiscardPkt64

Data Export Format

```
Version=2
collectionTime,collectorName,deviceName,ifIndex,policyName,classMapName,cbQosCMPrePolicyPk
t,cbQosCMPrePolicyByte,cbQosCMPrePolicyBitRate,cbQosCMPostPolicyByte,cbQosCMPostPolicyBitR
ate,cbQosCMDropPkt,cbQosCMDropByte,cbQosCMDropBitRate,cbQosCMNoBufDropPkt,cbQosPoliceConfo
rmedPkt,cbQosPoliceConformedByte,cbQosPoliceConformedBitRate,cbQosPoliceExceededPkt,cbQosP
oliceExceededByte,cbQosPoliceExceededBitRate,cbQosPoliceViolatedPkt,cbQosPoliceViolatedByt
e,cbQosPoliceViolatedBitRate,cbQosQueueingCurrentQDepth,cbQosQueueingMaxQDepth,
cbQosQueueingDiscardByte64,cbQosQueueingDiscardPkt64

2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,2,test3,class-default,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,2,test3,prec2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,2,test3,prec3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,34,test3,class-default,72010249855,72558379519,0,
72558379519,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,34,test3,prec2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,34,test3,prec3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,54,DT,class-default,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,54,DT,queue1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,54,DT,queue2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,54,DT,queue3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,55,DT,class-default,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,55,DT,queue1,72164241609,7212218179839,0,72831625
6227,0,72578850624,773406468183,0,0,0,0,0,0,0,0,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,55,DT,queue2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,55,DT,queue3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,56,test5,class-default,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,57,test5,class-default,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0
2002-11-11T22:40:05.000Z,CBQoS,cbqos-dev,58,test5,class-default,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0
```

ClusterMib Collector

The ClusterMib Collector collects statistics from multiple devices in a cluster and summarizes data for each cluster. ClusterMib Collector only takes **deviceGroup** instead of both device and deviceGroup. Each deviceGroup is essentially a cluster of devices. Data collected from the devices within a deviceGroup will be summarized and presented to management interface. Attributes will be summarized to sum, min, max, average.

XML Configuration

The <ClusterMib Collector> tag contains configuration specific to the ClusterMib Collector. It has a tag *snmpGet* that contains a list of OIDs. *snmpGet* uses SNMP get to poll the SNMP agent for an instance OID. The OIDs can be specified as the name of the node or in numeric form. For example system.sysDescr.0 can be specified as follows: *sysDescr.0, 1.1.0*, or *.1.3.6.1.2.1.1.1.0*.

Sample XML configuration (schedule and deviceGroup must be defined prior to collector):

```
<collector name="cluster_collection">
      <ClusterMibCollector>
            <schedule name="s1"/>
            <deviceGroup name="dg1"/>
            <snmpGet>
                  <oid>cvpdnSystemTunnelTotal</oid>
                  <oid>cvpdnSystemSessionTotal</oid>
                  <oid>cvpdnSystemDeniedUsersTotal</oid>
            </snmpGet>
      </ClusterMibCollector>
</collector>
```

Data Export

The ClusterMib Collector exports data in XML format. Data can be exported at a specified frequency or on-demand using the XML interface (<*get*> <*data*> tags).

Sample data:

```
<ClusterMibData>
      <collector name="cluster_collection">
            <collectionTime  time="2001-12-21T03:15:00Z">
                  <cluster name="dg1">
                        <device name="GW1.cisco.com"/>
                        <device name="GW2.cisco.com"/>
                        <attribute name="cvpdnSystemTunnelTotal">
                              <sum>5000</sum>
                              <min>1000</min>
                              <max>4000</max>
                              <avg>2500</avg>
                              <count>2</count>
                        </attribute>
                        <attribute name="cvpdnSystemSessionTotal">
                              <sum>10000</sum>
                              <min>3000</min>
                              <max>7000</max>
                              <avg>5000</avg>
                              <count>2</count>
                        </attribute>
                  </cluster>
            </collectionTime>
      </collector>
</ClusterMibData>
```

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The ClusterMib Collector supports threshold crossing/clearing alerts (TCAs). There are five different types of threshold attributes for each OID collected. These includes sum, min, max and avg of the OID values, and count for number of devices collected from. For example, with *<oid>cvpdnSystemDeniedUsersTotal.0</oid>*, the following attributes can be checked against threshold:

- sysUpTime.0.sum
- sysUpTime.0.min
- sysUpTime.0.max
- sysUpTime.0.avg
- sysUpTime.0.count

It supports purging of collected performance data. This can be done based on purge parameters in **das.properties** or you can configure a purger for the MIB Collector.

It supports hotspot polling. If the collector is configured for hotspot polling, Cisco CNS PerfE does not store the data internally and does not post-process the data after data collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

Cisco CNS PerfE supports data collection from any SNMP v2c MIBs. The MIBs should be loaded before the collector is configured in Cisco CNS PerfE.

Limitations

This collector supports SNMP v2c and SNMPv3.

No support for plugin.

CMNM Collector

The CMNM Collector works with a CEMF machine running the Cisco Media Gateway Center Node Manager (CMNM) to collect performance data about the devices managed by that element manager.

Requirements

The following criteria must be met to use the CMNM Collector:

- You must turn on CEMF performance polling using CMNM element manager (see Chapter 7 in **http://www.cisco.com/univercd/cc/td/doc/product/access/sc/rel8/cmnmgr/ramb15.pdf** for instructions.)
- There must be a Telnet account on the CMNM machine with permissions to run CEMF command line programs.
- There must be an FTP account on the CMNM machine.
- The Telnet account on the CMNM machine used by the Cisco CNS PerfE software must use **bash** as the default shell. The UNIX shells, sh, ksh, and csh, have a limitation on the command length.The **history Admin export** command length exceeds this limitation and hence the Cisco CNS PerfE software will not be able to execute this command. Use the **bash** shell to overcome this limitation.

XML Configuration

The <cmnmCollector> tag contains configuration information specific to only the CMNM Collector.

Table 6-3 Tags in the XML Configuration for the CMNM Collector

XML Tag	Purpose	Mandatory/ Optional
cemf-home	The path where CEMF was installed on the CEMF machine. The default value for this setting is **/opt/cemf**.	Optional
export-file	The prefix to the name of the files generated by CEMF. This prefix is concatenated with a timestamp to yield unique filenames for each collection to prevent problems if the generated files can not be cleaned up. CEMF will create files relative to **$CEMF_HOME/bin/.attributeHistoryServer.sysmgr**.	Optional
max-export-file-size	Limits the size of the files generated by CEMF. However, a value other than **-1** can result in CEMF generating multiple files per CMNM collection and the CMNM Collector does not process these secondary files.	Optional
command-delay	The delay (in seconds) that gives CEMF time to complete exporting CMNM performance data before the CMNM Collector tries to FTP the generated file.	Optional
history-criteria	Specifies which CEMF **historyCriteria** files will be part of the CEMF export during each CMNM collection. History criteria files contain sets of performance attributes collected by CEMF. There must be at least one history-criteria element per CMNM Collector.	Mandatory

Sample XML

```
<?xml version="1.0" encoding="UTF-8"?>
<das xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation='das.xsd'>

      <create>
            <!-- Collection schedule -->
            <schedule name="OneMinSchedule">
                  <start>2001-10-25T00:00:00.000-08:00</start>
                  <interval>PT1M</interval>
                  <interTaskDelay>PT0.1S</interTaskDelay>
            </schedule>

            <!-- Export schedule -->
            <schedule name="ThreeMinSchedule">
                  <start>2001-10-25T00:00:00.000-08:00</start>
                  <interval>PT3M</interval>
                  <interTaskDelay>PT0.1S</interTaskDelay>
            </schedule>

            <!-- CEMF machine -->
            <device name="cmnm-1.myco.com">
                  <timezone>-00:00</timezone>
                  <telnet>
```

```
                        <ipaddress>64.102.41.44</ipaddress>
                        <username>telnetUsername</username>
                        <password>telnetPassword</password>
                        <prompt>&gt;</prompt>
                  </telnet>
                  <ftp>
                        <ipaddress>64.102.41.44</ipaddress>
                        <username>ftpUsername</username>
                        <password>ftpPassword</password>
                  </ftp>
            </device>

            <!-- Attribute Threshold Crossing Alert -->
            <threshold name="thr1">
                  <!-- CEMF-specific attribute name -->
                  <attribute name="mgcController:RFC1213-MIB.udpOutDatagrams">
                        <raiseOperation>LT</raiseOperation>
                        <raiseValue>817200</raiseValue>
                        <clearOperation>GT</clearOperation>
                        <clearValue>817200</clearValue>
                        <level>critical</level>
                  </attribute>
            </threshold>

            <!-- FTP Data Handler -->
            <dataHandler name="ftpDH1">
                  <url>
                        <prefix>ftp://username:password@storage.myco.com/%2Fdata/cemf/</pr
efix>
                  </url>
            </dataHandler>

            <!-- CNS-PE Collector -->
            <collector name="CMNMTest">
                  <cmnmCollector>
                        <schedule name="OneMinSchedule"/>
                        <device name="cmnm-1.myco.com">
                              <threshold name="thr1"/>
                        </device>
                        <dataHandler name="ftpDH1">
                              <schedule name="ThreeMinSchedule"/>
                        </dataHandler>
                        <cemf-home>/opt/cemf</cemf-home>
                        <export-file>/export/dumpFile</export-file>
                        <max-export-file-size>-1</max-export-file-size>
                        <command-delay>30</command-delay>
                        <history-criteria>bamsChassisHistoryCriteria</history-criteria>
                        <history-criteria>hostAPCHistoryCriteria</history-criteria>
                  </cmnmCollector>
            </collector>
      </create>

      <start>
            <collector name="CMNMTest"/>
      </start>
</das>
```

Data Export

The CMNM Collector exports data in Comma Separated Values (CSV) format. The first line of each exported file contains the column headings separated by commas. Each subsequent line contains the following data:

Table 6-4 Order of the Data Exported by the CMNM Collector

Order	Parameter
1.	CMNM device
2.	CMNM-managed device
3.	data collection timestamp
4.	attribute name
5.	attribute value

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The CMNM Collector supports threshold crossing alerts (TCAs). TCAs are configured for specific attributes. The collector checks for TCAs after each collection of performance data.

Note Attribute names for TCAs are CEMF-specific.

The CMNM Collector supports purging of collected performance data. This is done on a schedule configured for all of Cisco CNS PerfE or based on an individual purger.

Limitations

Use the following information when you configure this collector:

- Files created by the CEMF command line programs have root permissions. In order to properly clean up these files, the <export-file> value should use an absolute path from which the Telnet account userid has permission to remove any file. (for example, **/tmp**). Any relative path used for this value will be relative to **$CEMF_HOME/bin/.attributeHistoryServer.sysmgr**, the default behavior of the CEMF command line application.
- If the <max-export-file-size> value is not **-1**, then there it is possible that CEMF will generate multiple files with different extensions per collection. The CMNM Collector does not process these secondary files.
- All CMNM device and attribute names are CEMF-specific.
- The Telnet prompt (<device><prompt>) must be set correctly to successfully login to the CEMF machine. If it is not correct, the collector will appear to hang. If you see messages in the Cisco CNS PerfE log files about CMNM Collector not completing its previous collection, make sure the Telnet prompt is set correctly.
- The <command-delay> setting allows the CEMF machine time to complete exporting performance data. This is the delay between executing the CEMF command line application to export data and the FTP transfer of the generated file. This value should be large enough so that the CEMF

application completes before the collector tries to FTP the generated file. The only true way to know if a value is safe is to observe the time it takes for the **$CEMF_HOME/bin/historyAdmin export** command to finish generating output.

- The CMNM Collector talks to the CEMF machine to get performance data, but the two machines may be in different time zones. If the CEMF machine is in a different time zone than the Cisco CNS PerfE server, it is important to set the time zone setting (<device><timezone>) for the CEMF machine. This will allow the CMNM Collector to properly adjust timestamps that do not have time zone values.

DOCSIS Data Collector

The Cisco CNS Performance Engine provides support for collecting DOCSIS information. This is achieved by using the DOCSIS Data Collector (DDC) included in Cisco CNS PerfE.

The DDC is designed to collect information that is of interest for a NOC operator at a Multiple System Operator (MSO), a cable service provider that also provides other services such as high-speed data (HSD) and/or voice telephony. The DDC provides information for the NOC personnel to examine the following information for a customer's cable modem:

- The cable modem's status (functioning or non-functioning)
- The status of other infrastructure equipment involved in providing high-speed data service.

The DDC interacts with the cable modem and the CMTS, which is Cisco's uBR, via SNMP and can use information from a customers's database for cable modem to CMTS association and to obtain location information.

The DOCSIS Data Collector collects data via SNMP and via ftp/cnsbus forwards information from multi-vendor CMTS's and the cable modems attached across the HFC plant.

Requirements

The Docsis Data Collector requires the framework definition of the CMTS's as a device. See XML configuration below.

If the number of CMTS's under management is more than a nominal number (10) it is recommended that you increase the Java property for max cache.

In the file **$DAS_HOME/config/das.properties** (at or around line 82) change

MEM.JAVA_MAX_CACHE=256M

to

MEM.JAVA_MAX_CACHE=512M

You may have to further increase the cache if the number of CMTS's under management exceeds 25.

XML Configuration

The following is a sample XML configuration for the DDC (schedule, device and data handlers must be defined prior to the collector). The database must contain an entry for **uBR01**.

Define the device to the framework as an SNMPv3 capable device:

```
<device name="ubr1">
  <snmp>
    <ipaddress>10.5.1.2</ipaddress>
    <readCommunity>public1</readCommunity>
    <timeout>PT5S</timeout>
    <retry>4</retry>
    <version>SNMPv3</version>
    <port>161</port>
    <userName>HORSHAM</userName>
    <authProtocol>MD5</authProtocol>
    <authPassword>HORSHAM1</authPassword>
  </snmp>
</device>
```

Define the collector:

```
<collector name="docsis1">
  <DocsisCollector>
    <schedule name="sch1M"/>
    <device name="ubr1">
      <threshold name="threshold1"/>
      <cmtsConfig>
```

Configure the cable modem on the uBR and define them to also be SNMPv3:

```
        <cmConfig>
          <readCommunity>public</readCommunity>
          <timeout>PT5S</timeout>
          <retry>1</retry>
          <version>SNMPv3</version>
          <port>161</port>
          <userName>HORSHAM</userName>
          <authProtocol>MD5</authProtocol>
          <authPassword>HORSHAM1</authPassword>
        </cmConfig>
        <row_origin>1</row_origin>
        <subscriber_count>0</subscriber_count>
        <threadPolicy>0</threadPolicy>

<serviceLostPercentageThreshold>10.0</serviceLostPercentageThreshold>
        <collectCmtsOnly>0</collectCmtsOnly>

        <collectExtraIfTable>1</collectExtraIfTable>
```

Note This is an AND function. If the **collectExtraIfTable** value is **1**, then extra information is collected. See the Data Export section below for information returned when this flag is set to **1**.

```
    <collectCdxQosCtrlUpTable>1</collectCdxQosCtrlUpTable>
  <collectCdxCmtsCmStatusTable>1</collectCdxCmtsCmStatusTable>
<collectDocsIfCmtsServiceTable>1</collectDocsIfCmtsServiceTable>
        <collectCdxCmCpeTable>1</collectCdxCmCpeTable>

        <throttleSleepMillis>0</throttleSleepMillis>
        <throttleBusyHours>
       <docsisBusyHours>1100-1300</docsisBusyHours>
       <docsisBusyHours>1430-1900</docsisBusyHours>
         <docsisBusyHours>2330-0210</docsisBusyHours>
```

```
            </throttleBusyHours>
          </cmtsConfig>
        </device>
        <device name="ubr2"/>
        <notifier name="CNSnotifier"/>
        <dataHandler name="hotspot"/>
        <dataHandler name="ddc-dh">
          <schedule name="sch2M"/>
        </dataHandler>
        <purger name="purge1"/>
        <globalConfig name="GlobalConfig1">
          <maxOfflinePercent>40.0</maxOfflinePercent>
          <lowSnrUpsteam>-25.0</lowSnrUpsteam>
         <upstreamXmitPowerFloor>34.0</upstreamXmitPowerFloor>
      <upstreamXmitPowerCeiling>34.0</upstreamXmitPowerCeiling>
          <downstreamSnrFloor>30.0</downstreamSnrFloor>
  <downstreamReceivePowerFloor>-15.0</downstreamReceivePowerFloor>
 <downstreamReceivePowerCeiling>5.0</downstreamReceivePowerCeiling>
          <maxModemsInitDCount>50</maxModemsInitDCount>
     <pctModemsOnlineFloor_Cmts>90.0</pctModemsOnlineFloor_Cmts>
         <minimumModemCount_Cmts>100</minimumModemCount_Cmts>
<pctModemsOnlineFloor_Upstream>90.0</pctModemsOnlineFloor_Upstream>
    <minimumModemCount_Upstream>100</minimumModemCount_Upstream>
       <pctModemsOnlineFloor_Ds>90.0</pctModemsOnlineFloor_Ds>
  <minimumModemCount_Downstream>100</minimumModemCount_Downstream>
     <pctModemsOnlineFloor_Node>90.0</pctModemsOnlineFloor_Node>
         <minimumModemCount_Node>100</minimumModemCount_Node>
       <writeCmtsReportsOnTimeout>1</writeCmtsReportsOnTimeout>
          <collectCmtsOnly>0</collectCmtsOnly>
          <maxDownstreamModems>1500</maxDownstreamModems>
        <maxMarketModemsThreading>20000</maxMarketModemsThreading>
        </globalConfig>
      </DocsisCollector>
    </collector>
```

Database Tables

The database tables used by the DOCSIS Data Collector must be defined after installation of Cisco CNS PerfE. They are not installed with the generic CNS PerfE release.

To define the DDC tables, do the following.

Step 1 Define the UNIX environment variable **DAS_HOME** pointing to the root directory where Cisco CNS PerfE was installed.

Step 2 Verify that the Cisco CNS PerfE system is running. If it is not, start CNS PerfE by entering the following:

cd $DAS_HOME/bin

./start.sh

Step 3 Change the directory to **$DAS_HOME/ddc/bin** and execute the following to create the tables that DOCSIS data collector will use:

./install-docsis-tables.sh

The DOCSIS Data Collector in CNS PerfE, Release 2.1 does not require the data in the tables to be pre-populated as in CNS PerfE, Release 2.0 but rather allows full configuration via XML.

Data Export, Threshold Crossing Alerts, Purging Data and Hotspot polling

The DOCSIS Data Collector fully supports historical data storage under control of the purge function of the framework for storage intervals defined.

The DDC supports data export forms of hotspot and periodic polling. Hotspot polling and can be configured via the XML interface.

Data Export

The following section contains a sample data export.

```
<DOCSIS_CHANNEL_DATA>
ipAddress|ifIndex|ifDescr|ifType|totalModems|onlineModems|snmpOnlineModems|snmpT
imeouts|lowXmitPowerExceptions|highXmitPowerExceptions|lowSnrExceptions|minXmitP
ower|maxXmitPower|avgXmitPower|minSnr|maxSnr|avgSnr|lowDsRecvPowerExceptions|hig
hDsRecvPowerExceptions|docsIfSigQSignalNoise|docsIfSigQUnerroreds|docsIfSigQCorr
ecteds|docsIfSigQUncorrectables|docsIfSigQMicroreflections|userExtensionData|snm
pPollDuration|lastUpdated
10.87.117.2|4|Cable3\/0-upstream0|129|0|0|0|0|0|0|0.00|0.00|0.00|0.00|0.00|0.0
0|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.616
10.87.117.2|5|Cable3\/0-upstream1|129|2|2|2|0|0|2|0|40.00|40.00|40.00|38.40|41.3
0|39.85|0|2|31.30|1414432329|354624783|3391950922|0| |86|2003-03-12 08:32:13.593
10.87.117.2|6|Cable3\/0-upstream2|129|0|0|0|0|0|0|0|0.00|0.00|0.00|0.00|0.00|0.0
0|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.57
10.87.117.2|7|Cable3\/0-upstream3|129|0|0|0|0|0|0|0|0.00|0.00|0.00|0.00|0.00|0.0
0|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.686
10.87.117.2|8|Cable3\/0-upstream4|129|0|0|0|0|0|0|0|0.00|0.00|0.00|0.00|0.00|0.0
0|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.64
10.87.117.2|9|Cable3\/0-upstream5|129|0|0|0|0|0|0|0|0.00|0.00|0.00|0.00|0.00|0.0
0|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.626
10.87.117.2|10|Cable3\/0-downstream|128|2|2|2|0|0|2|0|40.00|40.00|40.00|38.40|41
.30|39.85|0|2|0.00|0|0|0|0| |0|2003-03-12 08:32:13.7
10.87.117.2|14|Cable6\/0-upstream0|129|5|5|5|0|1|3|0|33.70|37.00|35.04|33.20|36.
60|35.22|0|0|28.90|6750785|0|5|0| |353|2003-03-12 08:32:13.603
10.87.117.2|15|Cable6\/0-upstream1|129|0|0|0|0|0|0|0|0.00|0.00|0.00|0.00|0.00|0.
00|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.58
10.87.117.2|16|Cable6\/0-downstream|128|5|5|5|0|1|3|0|33.70|37.00|35.04|33.20|36
.60|35.22|0|0|0.00|0|0|0|0| |0|2003-03-12 08:32:13.713
</DOCSIS_CHANNEL_DATA>
```

The following section provides raw data as retrieved from the attached cable modem.

```
<DOCSIS_MODEM_DATA>
docsIfCmtsCmStatusMacAddress|docsIfCmtsCmStatusIpAddress|cmtsIpAddress|sysDescr|
sysObjectID|sysUpTime|cmtsCmStatusDownChannelIndex|cmtsCmStatusUpChannelIndex|cm
tsCmStatusValue|cmtsCmStatusRxPower|cmtsCmStatusTimingOffSet|cmtsCmStatusIndex|d
ocsIfDownChannelPower|docsIfSigQUnerroreds|docsIfSigQUncorrectables|docsIfSigQSi
gnalNoise|docsIfCmStatusLostSyncs|docsIfCmStatusTxPower|docsIfCmStatusResets|doc
sIfCmStatusT1Timeouts|docsIfCmStatusT2Timeouts|docsIfCmStatusT3Timeouts|docsIfCm
StatusT4Timeouts|cdxIfCmtsCmStatusDynSidCount|cdxIfCmtsCmStatusAddlInfo|dot1dBas
ePortDlyExcdedDscards|docsDevServerConfigFile|cdxCmtsCmStatusValue|statusNotes|u
serExtensionData|lastUpdated|opticalNodeName
00:04:27:a5:c0:81|10.1.5.15|10.87.117.2|<<HW_REV: 3.t; VENDOR: Cisco Systems Inc
; BOOTR: 12.2(2r)T1; SW_REV: 12.2; MODEL: CVA122 >>IOS (tm) 120 Software
(CVA120-K8O3V4Y5-M), Experimental Version 12.2 Copyright (c) 1986-2002 by cisco Systems,
IncCompiled Sat 06-Apr-02 02:07 by
|.1.3.6.1.4.1.9.1.306|130.37|16|14|6|-.20|2810.00|393220|3.70|1215856659|8295|35.20|9|33.7
0|570|0|0|13|7|0|00 |2.0|HorshamCM.cm|12|Upstream Xmit Power 33.7 is below 34.0|
|2003-03-12 08:32:13.476|
00:02:16:d5:a4:0d|10.1.5.13|10.87.117.2|Cisco Internetwork Operating System Soft
ware IOS (tm) 920 Software (UBR920-K8V7Y5-M), Version 12.2(1), RELEASE SOFTWARE (fc2)
```

```
Copyright (c) 1986-2001 by cisco Systems, Inc.Compiled Fri 27-Apr-01 05:46 by cmong
|.1.3.6.1.4.1.9.1.255|56.95|16|14|6|-0.20|2813.00|393219|0.00|28728630|0|36.60|2|37.00|128
|0|0|38|2|0|00 |2.0|HorshamCM.cm|12|Upstream Xmit Power 37.0 is above 34.0| |2003-03-12
08:32:13.486|
08:00:3e:08:ef:4c|10.1.5.10|10.87.117.2|Motorola Docsis Cable Modem; MAC=08-00-3
E-08-EF-4C; SN=26529486; REV=2.0.0 |.1.3.6.1.4.1.161.7.10.5|1.74|16|14|6|-0.20|2
132.00|393218|0.00|426136520|0|33.20|0|34.00|0|0|0|0|0|0|00 |1.0| |12| | |2003-0
3-12 08:32:13.5|
00:08:0e:16:fb:be|10.1.4.51|10.87.117.2|<<HW_REV: 0; VENDOR: Motorola; BOOTR: CG
4D_05.3.04; SW_REV: CG4D_05.4.02; MODEL: SBV4200>>OS: VxWorks 5.4|.1.3.6.1.4.1.1
166.1.200.0.1.5.4.2|97.61|10|5|6|-0.20|652.00|196610|11.70|58160070|0|41.30|0|40
.00|0|0|0|4|1|0|00 |1.0|cisco11_perl_5304_cm_dual.bin|12|Upstream Xmit Power 40.
0 is above 34.0; Downstream Recv Power 11.7 is above 5.0| |2003-03-12 08:32:13.5
13|
</DOCSIS_MODEM_DATA>
```

If the flag **<collectExtraIfTable>1</collectExtraIfTable** is set, the following is also returned.

Note The values for **cdxQosCtrlUpMaxRsvdBWPercent**, **cdxQosCtrlUpAdmissionRejects**, and **cdxQosCtrlUpReservedBW** in the following data require the flag **<collectCdxQosCtrlUpTable>1</collectCdxQosCtrlUpTable>** to be set.

```
<DOCSIS_EXTRA_IF_LIST>
cmtsip|ifIndex|lastCollectedTime|ifType|ifAdminStatus|ifOperStatus|ifLastChange|
ifInOctets|ifInUcastPkts|ifInDiscards|ifInErrors|ifOutOctets|ifOutUcastPkts|ifOu
tDiscards|ifOutErrors|ifInMulticastPkts|ifInBroadcastPkts|ifOutMulticastPkts|ifO
utBroadcastPkts|ifCounterDiscontinuityTime|cdxCmtsCmTotal|cdxCmtsCmActive|cdxCmt
sCmRegistered|cdxQosCtrlUpMaxRsvdBWPercent|cdxQosCtrlUpAdmissionRejects|cdxQosCt
rlUpReservedBW
10.87.117.2|3|2003-03-12 08:32:13.146|127|1|1|959|2628718892|498464816|0|0|25280
32783|512134705|0|0|0|12466|608365|105159|0|3|3|3|null|null|null
10.87.117.2|4|2003-03-12 08:32:13.133|129|1|1|913|0|0|0|17742|0|0|0|0|0|0|0|0|
null|null|null|100|0|0
10.87.117.2|5|2003-03-12 08:32:13.11|129|1|1|927|3816738722|522883010|0|21085721
04|0|0|0|0|0|12466|0|0|0|null|null|null|100|0|0
10.87.117.2|6|2003-03-12 08:32:13.08|129|2|2|878|0|0|0|0|0|0|0|0|0|0|0|0|null|
null|null|100|0|0
10.87.117.2|7|2003-03-12 08:32:13.056|129|2|2|878|0|0|0|0|0|0|0|0|0|0|0|0|null
|null|null|100|0|0
10.87.117.2|8|2003-03-12 08:32:13.033|129|2|2|878|0|0|0|0|0|0|0|0|0|0|0|0|null
|null|null|100|0|0
10.87.117.2|9|2003-03-12 08:32:13.013|129|1|2|878|0|0|0|0|0|0|0|0|0|0|0|0|null
|null|null|100|0|0
10.87.117.2|10|2003-03-12 08:32:13.156|128|1|1|959|0|0|0|0|2528033883|512134710|
0|0|0|0|608365|105159|0|null|null|null|null|null|null
10.87.117.2|13|2003-03-12 08:32:13.123|127|1|1|351954390|270627401|999558|212|10
0|180696605|801821|0|0|24|43420|0|178151|0|8|8|8|null|null|null
10.87.117.2|14|2003-03-12 08:32:13.093|129|1|1|352003221|434415123|6957878|0|218
|0|0|0|0|24|43422|0|0|0|null|null|null|100|0|0
10.87.117.2|15|2003-03-12 08:32:13.066|129|2|2|351954093|0|0|0|0|0|0|0|0|0|0|0
|0|null|null|null|100|0|0
10.87.117.2|16|2003-03-12 08:32:13.043|128|1|1|351954390|0|0|0|0|180696605|80182
1|0|0|0|0|0|178151|0|null|null|null|null|null|null
</DOCSIS_EXTRA_IF_LIST>
```

If the flags **<collectExtraIfTable> = 1** and **<collectDocsIfCmtsServiceTable> = 1** are set, the following is also returned:

```
<DOCSIS_IF_CMTS_SERVICE_LIST>
cmtsip|ifIndex|docsIfCmtsServiceId|lastCollectedTime|docsIfCmtsServiceCmStatusIn
dex|docsIfCmtsServiceAdminStatus|docsIfCmtsServiceQosProfile|docsIfCmtsServiceCr
```

```
eateTime|docsIfCmtsServiceInOctets|docsIfCmtsServiceInPackets|cdxIfCmtsServiceOu
tOctets|cdxIfCmtsServiceOutPackets
10.87.117.2|3|1|2003-03-12 08:32:13.433|196609|1|0|2200|51440374|180004|0|0
10.87.117.2|3|2|2003-03-12 08:32:13.423|196610|1|0|2600|52730034|192288|0|0
10.87.117.2|3|3|2003-03-12 08:32:13.416|196610|1|0|2600|606988678|5057117|0|0
10.87.117.2|3|4|2003-03-12 08:32:13.41|196609|1|0|2200|604748942|5037968|0|0
10.87.117.2|3|4907|2003-03-12 08:32:13.406|196609|1|0|2200|34884|153|0|0
10.87.117.2|13|2|2003-03-12 08:32:13.393|393218|1|1|352004600|42775330|226849|29
112852|162147
10.87.117.2|13|3|2003-03-12 08:32:13.386|393219|1|1|352005600|43507260|121314|26
194529|120021
10.87.117.2|13|4|2003-03-12 08:32:13.38|393220|1|1|352007400|44949588|120993|261
97162|120033
10.87.117.2|13|5|2003-03-12 08:32:13.44|393221|1|1|352007600|45011810|123029|263
87483|121977
10.87.117.2|13|6|2003-03-12 08:32:13.43|393222|1|1|352080700|58213068|237276|267
81679|124815
</DOCSIS_IF_CMTS_SERVICE_LIST>
```

If the flags **<collectExtraIfTable> = 1** and **<collectCdxCmCpeTable> = 1** are set, the following is also returned:

```
<CDX_CM_CPE_LIST>
cmtsip|cdxCmCpeMacAddress|lastCollectedTime|cdxCmCpeType|cdxCmCpeIpAddress|cdxCm
CpeIfIndex|cdxCmCpeCmtsServiceId|cdxCmCpeCmStatusIndex
10.87.117.2|00:02:16:d5:a4:0d|2003-03-12 08:32:12.68|1|10.1.5.13|13|3|393219
10.87.117.2|00:02:16:d5:a4:27|2003-03-12 08:32:12.69|1|10.1.5.11|13|6|393222
10.87.117.2|00:04:27:a5:c0:45|2003-03-12 08:32:12.656|1|10.1.5.14|13|5|393221
10.87.117.2|00:04:27:a5:c0:81|2003-03-12 08:32:12.71|1|10.1.5.15|13|4|393220
10.87.117.2|00:04:bd:b7:81:ca|2003-03-12 08:32:12.696|1|10.1.4.50|3|1|196609
10.87.117.2|00:04:bd:b7:81:cc|2003-03-12 08:32:12.673|2|192.168.2.102|3|1|196609
10.87.117.2|00:08:0e:16:fb:be|2003-03-12 08:32:12.703|1|10.1.4.51|3|2|196610
10.87.117.2|00:08:0e:16:fb:c0|2003-03-12 08:32:12.686|2|192.168.2.104|3|2|196610
10.87.117.2|00:10:a4:aa:6e:fc|2003-03-12 08:32:12.723|2|192.168.3.60|13|5|393221
10.87.117.2|08:00:3e:08:ef:4c|2003-03-12 08:32:12.716|1|10.1.5.10|13|2|393218
</CDX_CM_CPE_LIST>
```

Pre-Defined Queries

In addition to the basic hotspot and periodic, the DOCSIS Data Collector also provides for the following pre-defined queries:

1. Query number of collection intervals currently available:

   ```
   <parameter name="docsisQueryType">
   docsis-number-collection-intervals
   </parameter>
   ```

2. Query of CMTS summary info:

   ```
   <parameter name="docsisQueryType">
   docsis-cmts-summary-info
   </parameter>
   ```

3. Query of CMTS upstream summary info:

   ```
   <parameter name="docsisQueryType">
   docsis-cmts-upstream-summary-info
   </parameter>
   ```

4. Query of specific upstream summary info by ifDescr:

   ```
   <parameter name="docsisQueryType">
   ```

```
docsis-specific-upstream-by-ifdescr-summary-info
</parameter>
<parameter name="ifDescr">
description
</parameter>
```

5. Query of specific upstream summary info by ifIndex:

```
<parameter name="docsisQueryType">
docsis-specific-upstream-by-ifindex-summary-info
</parameter>
<parameter name="ifIndex">
99
</parameter>
```

6. Query of all modem info:

```
<parameter name="docsisQueryType">
docsis-all-modem-info
</parameter>
```

7. Query of all modem info by upstream ifIndex:

```
<parameter name="docsisQueryType">
docsis-all-modem-info-by-upstream-ifindex
</parameter>
<parameter name="ifIndex">
99
</parameter>
```

8. Query of all modem info by optical node name:

```
<parameter name="docsisQueryType">
docsis-all-modem-info-by-optical-node-name
</parameter>
```

Threshold Crossing Alerts

The attributes listed in the following table are available for threshold crossing alerts processing or Process Plugins.

Note Threshold crossing alerts are limited to numeric data only so **sysDescr** would be invalid.

Table 6-5 Threshold Crossing Alert Attributes

Query String	tagname	Indexed By
sysUpTime	sysUpTime	None, scalar
sysDescr	sysDescr	None, scalar
sysObjectID	sysObjectID	None, Scalar
ifTable	ifDescr	X=ifIndex,
	ifType	
	docsIfSigQUnerroreds	
	docsIfSigQCorrecteds	

Table 6-5 Threshold Crossing Alert Attributes (continued)

Query String	tagname	Indexed By
	docsIfSigQUncorrectables	
	docsIfSigQSignalNoise	
	docsIfSigQMicroreflections	
	ifAdminStatus	
	ifOperStatus	
	ifLastChange	
	ifInOctets	
	ifInUcastPkts	
	ifInDiscards	
	ifInErrors	
	ifOutOctets	
	ifOutUcastPkts	
	ifOutDiscards	
	ifOutErrors	
ifXentry	ifCounterDiscontinuityTime	X=ifindex
	ifInMulticastPkts	
	ifInBroadcastPkts	
	ifOutMulticastPkts	
	ifOutBroadcastPkts	
docsIfCmtsServiceTable	docsIfCmtsServiceCmStatusIndex	X=ifIndex, Y=docsIfCmtsServicdeId
	docsIfCmtsServiceAdminStatus	
	docsIfCmtsServiceQosProfile	
	docsIfCmtsServiceCreateTime	
	docsIfCmtsServiceInOctets (related to the OutOctets in the next section)	
	docsIfCmtsServiceInPackets (related to the OutPackets in the next section)	
CdxCmtsServiceExtEntry	cdxIfCmtsServiceOutOctets	X=ifIndex Y= docsIfCmtsServiceId
	cdxIfCmtsServiceOutPackets	
cdxCmtsMacExtEntry	cdxCmtsCmTotal	X=ifIndex
	cdxCmtsCmActive	
	cdxCmtsCmRegistered	
cdxCmtsCmStatusExtEntry	cdxIfCmtsCmStatusDynSidCount	X=docsIfCmtsCmStatusIndex
	cdxIfCmtsCmStatusAddlInfo	
cdxCmCpeEntry	cdxCmCpeType	X=cdxCmCpeMacAddress

Table 6-5 Threshold Crossing Alert Attributes (continued)

Query String	tagname	Indexed By
	cdxCmCpeIfIndex	
	cdxCmCpeCmtsServiceId	
	cdxCmCpeIpAddress	
	cdxCmCpeStatusIndex	
cdxQosCtrlUpEntry	cdxQosCtrlUpMaxRsvdBWPercent	X=ifIndex
	cdxQosCtrlUpReservedBW	
	cdxQosCtrlUpAdmissionRejects	
cmtsSummaryData	ipaddress	None, scalars
	sysUpTime	
	collectionFinishTime	
	totalModems	
	onlineModems	
	snmpOnlineModems	
	cpeCount	
	lowXmitPowerExceptions	
	highXmitPowerExceptions	
	lowSnrExceptions	
	minXmitPower	
	avgXmitPower	
	minSnr	
	maxSnr	
	avgSnr	
	lowDsRecvPowerExceptions	
	highDsRecvPowerExceptions	
cmtsChannelData	ipAddress	X = ifIndex
	ifDescr	
	ifType	
	totalModems	
	onlineModems	
	snmpOnlineModems	
	lowXmitPowerExceptions	
	highXmitPowerExceptions	
	lowSnrExceptions	
	minXmitPower	
	avgXmitPower	
	minSnr	

Table 6-5 Threshold Crossing Alert Attributes (continued)

Query String	tagname	Indexed By
	maxSnr	
	avgSnr	
	lowDsRecvPowerExceptions	
	highDsRecvPowerExceptions	
	docsIfSigQSignalNoise	
	docsIfSigQUnerroreds	
	docsIfSigQCorrecteds	
	docsIfSigQUncorrectables	
	lastUpdated	

Note the following:

- For summary info on the CMTS, the names will be **cmtsSummaryData.<tagname>**
- For linecard data dealing with signal quality, the names will be **ifTable.<ifIndex>.<tagname>**
- For individual linecard data and modem info, the names will be **cmtsChannelData.<ifIndex>.<tagname>**
- For multiply index entries, the names will be **docsIfCmtsServiceTable.<ifIndex>.<docsIfCmtsServicdeId>.tagname**
- The threshold attribute name **ifTable.10.ifType** matches **ifType** values for **ifIndex** whose id is set to **10**.
- The threshold attribute name **ifTable..docsIfSigQCorrecteds** matches **docsIfSigQCorrecteds** values for all **ifIndex** files.

Purging Data

The Docsis Data Collector fully supports the framework purge mechanism.

IOSCLI Collector

The IOSCLI Collector issues a show command via telnet to a router and extracts information from output text of the show command. A regular expression file is associated with each show command. This regular expression file determines which fields of the show command output text to extract the data from. One must write the regular expression file for the show command.

Note The IOSCLI Collector is deprecated. It is recommended that you use the IosCliOps Collector. Refer to "IosCliOps Collector" section on page 6-46.

Note DDC does not support the definition of threshold parameters through the XML interface like the rest of the system. However, it can send TCAs similar to the rest of the system (as per the definition of notifier).

XML configuration

```
<das>
      <create>
            <schedule name="5min">
                  <interval>PT5M</interval>
            </schedule>

            <schedule name="15min">
                  <interval>PT15M</interval>
            </schedule>

            <device name="router">
                  <telnet>
                        <ipaddress>10.0.0.1</ipaddress>
                        <username>user1</username>
                        <password>pass1</password>
                  </telnet>
                  <ios>
                        <ipaddress>10.0.0.1</ipaddress>
                        <username>user1</username>
                        <password>pass1</password>
                        <secret>pass2</secret>
                        <terminalServerIpAddress>10.0.0.2</terminalServerIpAddress>
                        <terminalServerPort>2001</terminalServerPort>
                        <consoleLinePassword>pass3</consoleLinePassword>
                        <connectMode>terminalserver</connectMode>
                        <transportMode>telnet</transportMode>
                  </ios>
            </device>

            <dataHandler name="IosCliExporter">
                  <url>
                        <prefix>ftp://user1:pass1@10.0.0.1/%2Fdata/</prefix>
                  </url>
            </dataHandler>
            <notifier name="notifier1">
                  <cns>
                        <subject>cns.cisco.das.listener</subject>
                  </cns>
            </notifier>

            <threshold name="IosCliThresholds">
                  <attribute name="RAS.GK discovery.requests.sent">
                        <raiseOperation>EQ</raiseOperation>
                        <raiseValue>8</raiseValue>
                        <clearOperation>GT</clearOperation>
                        <clearValue>4</clearValue>
                        <level>minor</level>
                  </attribute>
            </threshold>


            <collector name="IosCliCollector">
                  <schedule name="5min"/>
                  <device name="router"/>
                  <threshold name="IosCliThresholds"/>
                  <notifier name="notifier1"/>
                  <dataHandler name="IosCliExporter">
                        <schedule name="15min"/>
                  </dataHandler>
                  <IosCliCollector>
                        <command>show h323 gateway</command>
```

```
                        <regexp-url>file:///cnsperfe/config/h323.gateway.re</regexp-url>
                  </IosCliCollector>
            </collector>

      </create>
      <add/>
      <start>
            <collector name="IosCliCollector"/>
      </start>
</das>
```

Note It is recommended that all the regular expression files be kept in the ***<CNS_PerfE_Home>*/config/ioscli** directory.

The **<transportMode>** parameter can be either Telnet or SSH depending on the login mode supported by device.

Note This collector only supports 3DES encryption when the SSH option is set for **<transportMode>**.

Data Export

The actual header attrs will depend on the show command and the format of the regular expression file. See the examples dir for a complete example of the regular expression file.

```
<SHOWCMD>
<HEADER>Timestamp,H225.request.setup.sent,H225.request.setup.rvcd,H225.request.s
etup.failed,H225.request.setup confirm.sent,H225.request.setup confirm.rcvd,H225
.request.setup confirm.failed,H225.request.Alert.sent,H225.request.Alert.rcvd,H2
25.request.Alert.failed,H225.request.Progress.sent,H225.request.Progress.rcvd,H2
25.request.Progress.failed,H225.request.Call proceeding.sent,H225.request.Call p
roceeding.rcvd,H225.request.Call proceeding.failed,H225.request.Notify.sent,H225
.request.Notify.rcvd,H225.request.Notify.failed,H225.request.Info.sent,H225.requ
est.Info.rcvd,H225.request.Info.failed,H225.request.User info.sent,H225.request.
User info.rcvd,H255.request.User info.failed,H225.request.Facility.sent,H225.req
uest.Facility.rcvd,H225.request.Facility.failed,H225.request.Release.sent,H225.r
equest.Release.rcvd,H225.request.Release.failed,H225.request.Reject.sent,H225.re
quest.Reject.rcvd,H225.request.Reject.failed,H225.request.Passthrough.sent,H225.
request.Passthrough.rcvd,H225.request.Passthrough.failed,H225.established timeou
t,RAS failed,H245 failed,RAS.GK discovery.requests.sent,RAS.GK discovery.confirm
s.rcvd,RAS.GK discovery.rejects.rcvd,RAS.Registration.requests.sent,RAS.Registra
tion.confirms.rcvd,RAS.Registration.rejects.rcvd,RAS.Admission.requests.sent,RAS
.Admission.confirms.rcvd,RAS.Admission.rejects.rcvd,RAS.Bandwidth.requests.sent,
RAS.Bandwidth.confirms.rcvd,RAS.Bandwidth.rejects.rcvd,RAS.Disengage.requests.se
nt,RAS.Disengage.confirms.rcvd,RAS.Disengage.rejects.rcvd,RAS.Unregister.request
s.sent,RAS.Unregister.confirms.rcvd,RAS.Unregister.rejects.rcvd,RAS.Resource ava
il.requests.sent,RAS.Resource avail.confirms.rcvd,RAS.Req in progress.requests.s
ent,RAS.GK discovery.requests.rcvd,RAS.GK discovery.confirms.sent,RAS.GK discove
ry.rejects.sent,RAS.Registration.requests.rcvd,RAS.Registration.confirms.sent,RA
S.Registration.rejects.sent,RAS.Admission.requests.rcvd,RAS.Admission.confirms.s
ent,RAS.Admission.rejects.sent,RAS.Bandwidth.requests.rcvd,RAS.Bandwidth.confirm
s.sent,RAS.Bandwidth.rejects.sent,RAS.Disengage.requests.rcvd,RAS.Disengage.conf
irms.sent,RAS.Disengage.rejects.sent,RAS.Unregister.requests.rcvd,RAS.Unregister
.confirms.sent,RAS.Unregister.rejects.sent,RAS.Resource avail.requests.rcvd,RAS.
Resource avail.confirms.sent,RAS.Req in progress.requests.rcvd</HEADER>
<DATA>2002-10-02 15:02:48.233,11,0,0,0,11,0,0,0,0,0,0,0,11,0,0,0,0,0,0,0,0,0,0
,44,44,0,11,11,0,0,0,0,0,0,0,0,0,0,482179,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0</DATA>
```

```
</SHOWCMD>
```

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The collector supports thresholds. The threshold attribute name format is as follows:

```
<attr-name>
```

For example:

```
RAS.GK discovery.requests.sent
```

See the XML configuration for a complete example (above).

The IOSCLI Collector supports purging data. It does not support hotspot polling.

IosCliOps Collector

The IosCliOps Collector is a collector supported in CNS PerfE, Version 2.1. This collector is used to poll information from IOS devices. It issues an IOS show command via telnet/ssh to a router and extracts information from output text of the show command. A regular expression file is associated with each show command. This regular expression file specifies the fields to be extracted from the show command output.

The IosCliOps Collector supports execution of multiple commands for a device. Each command is treated as a single operation, the command output is collected, and the regular expression is applied to extract the necessary attributes. The collected data is then stored in the persistent store, thresholds can be applied, and data can be exported.

You must provide the regular expression to parse the output of the command. Regular Expressions are placed in the following directory:

<CNS-PerfEHOME>/config/ioscli/

XML Configuration

```
<das>
    <create>
        <schedule name="5min">
            <interval>PT5M</interval>
        </schedule>

        <schedule name="15min">
        <interval>PT15M</interval>
        </schedule>

        <device name="dev1">
      <ios>
        <ipaddress>10.0.0.1</ipaddress>
                <username>user1</username>
                <password>pass1</password>
        <secret>pass2</secret>
        <terminalServerIpAddress>10.0.0.2</terminalServerIpAddress>
        <terminalServerPort>2001</terminalServerPort>
        <consoleLinePassword>pass3</consoleLinePassword>
```

```
        <connectMode>terminalserver</connectMode>
        <transportMode>telnet</transportMode>
      </ios>
        </device>

    <dataHandler name="dh1">
            <url>
        <prefix>ftp://user1:pass1@10.0.0.1/data/ios</prefix>
        </url>
    </dataHandler>
    <notifier name="n1">
        <cns>
           <subject>cns.cisco.das.listener</subject>
        </cns>
    </notifier>

        <threshold name="t1">
        <attribute name="ping.minRoundTripTime">
        <raiseOperation>EQ</raiseOperation>
        <raiseValue>20</raiseValue>
        <clearOperation>GT</clearOperation>
        <clearValue>100</clearValue>
        <level>minor</level>
        </attribute>
    </threshold>

        <collector name="c1">
          <IosCliOpsCollector>
              <schedule name="5min"/>
              <device name="dev1">
    <operation name="op1">
    <command>ping mpls ipv4 10.131.159.252/32 sweep 1492 1500 1</command>
    </operation>
     <operation name="op2">
        <command>ping mpls ipv4 10.131.159.252/32</command>
     </operation>
          </device>
              <threshold name="t1"/>
              <notifier name="n1"/>
              <dataHandler name="dh1">
        <schedule name="15min"/>
          </dataHandler>
              <regexp-url>file:///cnsperfe/config/ioscli/LspPing.re</regexp-url>
            </IosCliOpsCollector>
        </collector>
    </create>
    <start>
        <collector name="c1"/>
    </start>
</das>
```

The **<connectMode>** parameter of IOS device configuration can be either **terminalserver** or **direct** depending on the mode of device access from CNS PerfE.

The **<transportMode>** parameter can be either Telnet or SSH depending on the login mode supported by device.

Note This collector only supports 3DES encryption when the SSH option is set for **<transportMode>**.

Syntax of Regular Expression File

Regular expression file contains a pair of **<ATTRNAMES>** and **<REGEXP>** tags.

<ATTRNAMES> –Specifies the names for the attributes extracted from the command output.

<REGEXP> – Specifies the regular expression to be applied for the command. Contents to be extracted are embedded within common braces, for example:

```
.*Success rate is (\d+) percent
```

The number of attributes mentioned in the **<ATTRNAMES>** and the number of braces in the **<REGEXP>** tag should match. For example, the following is the regular expression file for Ping command.

```
<ATTRNAMES>ping.successRate,ping.packetsReceived,ping.packetsSent,ping.minRou
ndTripTime,ping.avgRoundTripTime,ping.maxRoundTripTime</ATTRNAMES>
<REGEXP>.*Success rate is (\d+) percent \((\d+)\/(\d+)\)(?:,\s+round-
trip\s+min\/avg\/max\s+=\s+(\d+)\/(\d+)\/(\d+) ms|$)</REGEXP>
```

Data Export Format

```
Version = 1.0
CollectorName, DeviceName, OperationName, CollectionTime,
ping.successRate,ping.packetsReceived,ping.packetsSent,ping.minRoundTripTime,ping.a
vgRoundTripTime,ping.maxRoundTripTime
c1,dev1,op2,2003-07-25T12:03:00.000Z,100,1,1,80,80,80
```

The header in the exported file contains the name of collector, device, operation name and collectionTime in addition to the attribute names specified in the regular expression file.

For a collector, when dataHandler is associated with Schedule, one file per device will be exported and this contains all the operations specified for this device.

For a collector, when **ConsolidatedExport** option is enabled, one file for all the devices, (and all the operations of the devices) associated with collector, will be exported.

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The collector supports thresholds. The threshold attribute name format is as follows:

```
<attr-name>
```

For example:

```
ping.maxRoundTripTime
```

See the XML configuration for a complete example (above).

The IosCliOps Collector supports purging data. It also supports hotspot polling. When hotspot polling is configured, one file will be exported for each operation associated with the device.

Add/Remove operations from device

Once the collector is created, operations can be added or removed dynamically as follows:

XML example for adding operations for a given device:

```
<das>
  <add>
    <collector name="c1">
       <IosCliOpsCollector>
          <device name="dev1">
     <operation name="op3">
      <command>ping mpls ipv4 10.131.159.252/32 sweep 1492 1500 1</command>
    </operation>
          </device>
       </IosCliOpsCollector>
    </collector>
  </add>
</das>
```

XML example for removing operations for a given device:

```
<das>
  <remove>
    <collector name="c1">
       <IosCliOpsCollector>
          <device name="dev1">
       <operation name="op3"/>
      </device>
       </IosCliOpsCollector>
    </collector>
  </remove>
</das>
```

Add/Remove operations from deviceGroup

Multiple operations can be specified for each device group. i.e. for each operation, CNS PerfE issues the IOS command at the router, collects the output, and extracts the necessary according to the regular expression file specified; this process is repeated for all the devices in the group.

```
<das>
    <create>
        <deviceGroup name="dg1">
      <device name="dev1"/>
      <device name="dev2"/>
        </deviceGroup>

        <collector name="c1">
          <IosCliOpsCollector>
              <schedule name="5min"/>
              <deviceGroup name="dg1">
       <operation name="op1">
       <command>ping mpls ipv4 10.131.159.252/32 sweep 1492 1500 1</command>
     </operation>
     <operation name="op2">
        <command>ping mpls ipv4 10.131.159.252/32</command>
     </operation>
          </deviceGroup>
              <dataHandler name="dh1">
       <schedule name="15min"/>
          </dataHandler>
              <regexp-url>file:///cnsperfe/config/ioscli/LspPing.re</regexp-url>
```

```
            </IosCliOpsCollector>
         </collector>

    </create>
    <start>
      <collector name="c1"/>
    </start>
</das>
```

XML example for adding operations for a given device group:

```
<das>
  <add>
    <collector name="c1">
       <IosCliOpsCollector>
          <deviceGroup name="dg1">
        <operation name="op3">
       <command>ping mpls ipv4 10.131.159.252/32 sweep 1492 1500 1</command>
    </operation>
      </deviceGroup>
       </IosCliOpsCollector>
    </collector>
  </add>
</das>
```

XML example for removing operations for a given device group:

```
<das>
  <remove>
    <collector name="c1">
       <IosCliOpsCollector>
          <deviceGroup name="dg1">
        <operation name="op3"/>
      </deviceGroup>
       </IosCliOpsCollector>
    </collector>
  </remove>
</das>
```

Limitations

This collector has a limitation that the same regular expression will be applied to all the command outputs in a given collector

No plugins are supported in CNS PerfE, version 2.1 for this collector.

Comparison with the IosCli Collector of CNS PerfE, Version 2.0

In CNS-PerfE 2.0, the IosCli Collector supports only one command per device and only one regular expression is allowed per collector. The IosCliOps Collector supports multiple operations per device and a single collector can support multiple devices (with the restriction that the same regular expression will be applied to all the command outputs in a given collector). The IosCliOps Collector can support either Telnet or SSH.

MIB Collector

This collector is used for generic data collection from SNMP agents. The MIB Collector supports individual MIB OID instance retrieval with SNMP get or retrieval of columns in a table with SNMP getbulk. It is recommended that SNMP getbulk be used when retrieving a large volume of data from one table or multiple tables for better performance. In order for the MIB Collector to collect the data, the relative MIB should be loaded first.

The following properties can be used to tune the performance of the MIB Collector:

- snmp.attrsPerPDU
- snmp.maxRepetitions
- snmp.interval

Requirements

The appropriate MIB has to be loaded before querying the SNMP attributes.

The following MIBS are automatically loaded when Cisco CNS PerfE software starts up but these MIBs will not show up in the XML configuration file:

- IANAifType-MIB
- RFC1213-MIB
- IF-MIB
- CISCO-SMI
- SNMPv2-MIB
- CISCO-RTTMON-MIB

XML Configuration

The <MibCollector> tag contains configuration specific to the MIB Collector. There are two child nodes: **snmpGet** and **snmpGetMany**. Either tag can contain a list of OIDs. **SnmpGet** uses SNMP **get** to poll the SNMP agent for an instance OID whereas **snmpGetMany** uses **getbulk** to poll the SNMP agent for columns. The OIDs can be specified as the name of the node or in numeric form. For example **system.sysDescr.0** can be specified as follows: **sysDescr.0**, **1.1.0**, or **.1.3.6.1.2.1.1.1.0**.

Sample XML configuration 1 (schedule must be defined prior to collector):

```
<collector name="mc">
      <MibCollector>
            <schedule name="sample1" />
            <snmpGet>
                  <oid>sysDescr.0</oid>
            </snmpGet>
            <snmpGetMany>
                  <oid>ifMtu</oid>
                  <oid>ifSpeed.10</oid>
            </snmpGetMany>
      </MibCollector>
</collector>
```

Table Retrieval

In addition to **<snmpGet>** and **<snmpGetMany>** tags, the MIB Collector also supports the **<snmpGetTable>** tag. Currently, the MIB Collector supports bulk retrieval using **<snmpGetMany>**. This mode of collection is efficient for large tables, but inefficient for small tables.

The **<snmpGetTable>** option can be used to retrieve small tables efficiently.

```
<collector name="mc">
      <schedule name="sample1" />
      <MibCollector>
            <snmpGet>
                  <oid>sysDescr.0</oid>
            </snmpGet>
            <snmpGetTable>
                  <table oid="ifTable">
                        <oid>ifInOctets</oid>
                        <oid>ifOutOctets</oid>
                  </table>
                  <table oid="atTable"/>
            </snmpGetTable>
      </MibCollector>
</collector>
```

The above example retrieves all the values for ifInOctets and ifOutOctets from ifTable and it retrieves the entire table for atTable.

Selective Polling

The selective polling feature allows you to poll specific oids which satisfy some criteria. The condition tags determine the match criteria. The first collection poll retrieves all the entries in the table for condition OIDs. In the consecutive polls, only the entries that match the condition criteria are retrieved. The **<where>** tag allows you to combine several conditions together using logical operators such as AND and OR.

Select the selective polling feature by specifying **<snmpGetTableQuery>** in the MibCollector section of the configuration.

<select>: Specifies the condition on which the OIDs will be polled.

<condition>: Specifies the condition. Supported operations are:

eq: equals

lt: less than

gt: greater than

<where>: Specifies the relationship between different **<condition>** tags. This tag is optional if only one condition exists.

Example 6-1 Fetch ifInOctets and ifOutOctets oids for all ethernetCsmacd(6) interfaces

```
<collector name="MibCollect1">
   <MibCollector>
       <schedule name="MibSchedule"/>
       <device name="MibDevice"/>
       <dataHandler name="MibExport"/>
       <snmpGet>
           <oid>sysName.0</oid>
       </snmpGet>
       <snmpGetTableQuery>
           <table oid="ifTable">
               <select>
                   <condition name="c1" oid="ifType" oper="eq">6</condition>
                   <where>c1</where>
               </select>
               <oid>ifInOctets</oid>
               <oid>ifOutOctets</oid>
           </table>
       </snmpGetTableQuery>
   </MibCollector>
</collector>
```

Example 6-2 Fetch ifInOctets and ifOutOctets oids for all interfaces where ifType equals ethernetCsmacd(6), ifDescr equals FastEthernet and ifMtu equals to 1500

```
<collector name="MibCollect1">
    <MibCollector>
        <schedule name="MibSchedule"/>
        <device name="MibDevice"/>
        <dataHandler name="MibExport"/>
        <snmpGet>
            <oid>sysName.0</oid>
        </snmpGet>
        <snmpGetTableQuery>
            <table oid="ifTable">
                <select>
                    <condition name="c1" oid="ifDescr" oper="eq">FastEthernet</condition>
                    <condition name="c2" oid="ifType" oper="eq">6</condition>
                    <condition name="c3" oid="ifMtu" oper="eq">1500</condition>
                    <where>c1 AND c2 OR c3</where>
                </select>
                <oid>ifInOctets</oid>
                <oid>ifOutOctets</oid>
            </table>
        </snmpGetTableQuery>
    </MibCollector>
</collector>
```

Example 6-3 Multiple table entries in <snmpGetTableQuery>

```
<collector name="MibCollect1">
    <MibCollector>
    …
        <snmpGetTableQuery>
            <table oid="ifTable">
                <select>
                    <condition name="c1" oid="ifDescr" oper="eq">FastEthernet</condition>
                    <condition name="c2" oid="ifType" oper="eq">6</condition>
                    <condition name="c3" oid="ifMtu" oper="eq">1500</condition>
                    <where>c1 AND c2 OR c3</where>
                </select>
                <oid>ifInOctets</oid>
                <oid>ifOutOctets</oid>
            </table>
            <table oid="ipRouteTable">
                <select>
                    <condition name="c1" oid="ipRouteAge" oper="eq">0</condition>
                    <where>c1</where>
                </select>
                <oid>ipRouteMetric1</oid>
                <oid>ipRouteMetric2</oid>
                <oid>ipRouteMetric3</oid>
                <oid>ipRouteMetric5</oid>
            </table>
        </snmpGetTableQuery>
    </MibCollector>
</collector>
```

Example 6-4 Multiple <snmpGetTableQuery> tag in MibCollector

```
<collector name="MibCollect1">
   <MibCollector>
   …
       <snmpGetTableQuery>
           <table oid="ifTable">
               <select>
                   <condition name="c1" oid="ifDescr" oper="eq">FastEthernet</condition>
                   <condition name="c2" oid="ifType" oper="eq">6</condition>
                   <condition name="c3" oid="ifMtu" oper="eq">1500</condition>
                   <where>c1 AND c2 OR c3</where>
               </select>
               <oid>ifInOctets</oid>
               <oid>ifOutOctets</oid>
           </table>
       </snmpGetTableQuery>
       <snmpGetTableQuery>
       <table oid="ipRouteTable">
           <select>
               <condition name="c1" oid="ipRouteAge" oper="eq">0</condition>
               <where>c1</where>
           </select>
           <oid>ipRouteMetric1</oid>
           <oid>ipRouteMetric2</oid>
           <oid>ipRouteMetric3</oid>
           <oid>ipRouteMetric5</oid>
   </table>
   </snmpGetTableQuery>
   </MibCollector>
</collector>
```

Note The **<where>** tag is not necessary when only one condition statement is specified.

Example 6-5 Complete selective polling

```
<das>
<load>
   <mib name="IF-MIB.my"/>
</load>
<create>
<collector name="MibCollect2">
<MibCollector>
   <schedule name="MibSchedule"/>
   <device name="MibDevice"/>
   <dataHandler name="MibExport"/>
   <snmpGet>
       <oid>sysName.0</oid>
   </snmpGet>
   <snmpGetTableQuery>
       <table oid="ifTable">
           <select>
               <condition name=”c1” oid=”ifType” oper=”eq”>6</condition>
               <where>c1 </where>
           </select>
               <oid>ifAdminStatus</oid>
               <oid>ifCounterDiscontinuityTime</oid>
               <oid>ifHCInBroadcastPkts</oid>
               <oid>ifHCInMulticastPkts</oid>
               <oid>ifHCInOctets</oid>
               <oid>ifHCInUcastPkts</oid>
               <oid>ifHCOutBroadcastPkts</oid>
               <oid>ifHCOutMulticastPkts</oid>
               <oid>ifHCOutOctets</oid>
               <oid>ifHCOutUcastPkts</oid>
               <oid>ifInErrors</oid>
               <oid>ifInUnknownProtos</oid>
               <oid>ifOperStatus</oid>
           </table>
       </snmpGetTableQuery>
   </MibCollector>
   </collector>
</create>
</das>
```

Example 6-6 Selective polling from the Memory Pool table

```
<das>
<load>
<mib name="CISCO-MEMORY-POOL-MIB.my"/>
</load>
<create>
<collector name="MibCollect3">
    <MibCollector>
        <schedule name="MibSchedule"/>
        <device name="MibDevice"/>
        <dataHandler name="MibExport"/>
        <snmpGet>
            <oid>sysName.0</oid>
        </snmpGet>
        <snmpGetTableQuery>
            <table oid="ciscoMemoryPoolTable">
                <select>
                    <condition name="c1"
                    oid="ciscoMemoryPoolName" oper="eq">Processor</condition>
                    <where>c1 </where>
                    </select>
                    <oid>ciscoMemoryPoolFree</oid>
                    <oid>ciscoMemoryPoolLargestFree</oid>
                <oid>ciscoMemoryPoolUsed</oid>
                <oid>ciscoMemoryPoolValid</oid>
            </table>
        </snmpGetTableQuery>
    </MibCollector>
    </collector>
</create>
</das>
```

Example 6-7 Selective polling from the cardTable

```
<das>
<load>
<mib name="OLD-CISCO-CHASSIS-MIB.my"/>
</load>
<create>
<collector name="MibCollect4">
    <MibCollector>
        <schedule name="MibSchedule"/>
        <device name="MibDevice"/>
        <dataHandler name="MibExport"/>
            <snmpGet>
                <oid>sysName.0</oid>
            </snmpGet>
        <snmpGetTableQuery>
            <table oid="cardTable">
                <select>
                    <condition name="c1" oid="chassisType" oper="eq">365</condition>
                    <where>c1 </where>
                </select>
                <oid>cardType</oid>
                <oid>cardDescr</oid>
                    <oid>cardIndex</oid>
                    <oid>cardSerial</oid>
                    <oid>cardHwVersion</oid>
                <oid>cardSwVersion</oid>
                    <oid>cardSlotNumber</oid>
                    <oid>cardContainedByIndex</oid>
                    <oid>cardOperStatus</oid>
                    <oid>cardSlots</oid>
                </table>
            </snmpGetTableQuery>
        </MibCollector>
    </collector>
</create>
</das>
```

Data Export

The MIB Collector exports data in Comma Separated Values (CSV) format. Data can be exported at specified frequency or on-demand using the XML interface (<get> <data> tags). The first line of each exported file contains the column headings separated by commas. Each subsequent line contains the following data:

Table 6-6 Order of the Data Exported by the MIB Collector

Order	Parameter
1.	collector name
2.	device IP address
3.	data collection timestamp
4.	attribute name
5.	attribute value

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The MIB Collector supports threshold crossing/clearing alerts (TCAs). TCAs are configured for specific attributes. The collector checks for TCAs after each collection period. Attribute names can include instance in the OID or just the column name. Examples of attribute names are: ifMtu, ifSpeed.10, ifMtu.15 etc.

The MIB Collector supports purging of collected performance data. This can be done based on purge parameters in **das.properties** or you can configure a purger for the MIB Collector.

The MIB Collector supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not store the data internally and does not post-process the data after collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

Cisco CNS PerfE supports data collection from any SNMP v2c and SNMP v3 MIBs. The MIBs should be loaded before the collector is configured in Cisco CNS PerfE.

Limitations

None.

MIB Collector Plugins

The MIB Collector supports two types of plugins, processor plugins and formatter plugins:

Processor Plugin: Processor plugins are used to perform computation on the collected data. Instead of exporting raw data, the collector can calculate link utilization and export this data, providing a real picture of the network.

Formatter Plugin: Formatter plugins can be used to format the collected data, which will provide additional attribute type information. When configured with the Formatter plugin, the export file of the MIB Collector will contain the type of the MIB attribute in addition to its default output.

Processor Plugin

Plugins are intended to perform computations on the collected data. For example, assume one would have collected ifInOctets and ifOutOctets using MIB Collector. Instead of exporting raw data, one can do some computation say calculate link utilization using the above collected data and export that data which gives real picture of the network.

This can be achieved by associating different processor components with the MIB Collector.

These processor components are designed to work with MIB Collector. Following are the conventions used in providing parameter names for computation.

```
<parameter name="<operation required>.<index>">attribute</parameter>
```

where

- <*operation required*> can be delta, int, var, op and result.
- <*index*> is an integer

Example:

1. <parameter name="delta.1">ifInOctets</parameter>

Delta Component means that delta operation has to be carried out on the parameter ifInOctets

2. <parameter name="var.1">ifSpeed</parameter>

Var component means that the value of attribute specified i.e., ifSpeed is to be used directly for computation.

3. <parameter name="int.1">100</parameter>

Int component is used to provide integer values for computation.

4. <parameter name="op.1">divide delta.2 int.1</parameter>

Op component for arithmetic operations.

First is operation and next two arguments are operands.

Operations supported are plus, minus, multiply and divide.

5. <parameter name="result.1">deltaIfInOctets delta.1</parameter>

Result component is to save and export the computed data.

Computed data is stored and exported with the name "deltaIfInOctets".

Data saved and also exported is delta.1, which is nothing but delta of ifInOctets.

Note that only two operands are allowed for each operation. If the result of a Op Component is a scalar value, then this component value must be used as a second operand in another Op Component. For example:

```
<parameter name="delta.1">sysUpTime.0</parameter>
<parameter name="var.1">ifSpeed</parameter>
<parameter name="int.1">100</parameter>
<parameter name="op.1">divide delta.1 int.1</parameter>
<parameter name="op.2">multiply op.1 var.1</parameter>
```

In this example configuration, **op.1** yields a scalar value. As a result, **op.1** cannot be used as first operand in the **op.2** calculation. It has to be used as follows:

```
<parameter name="op.2">multiply var.1 op.1</parameter>
```

In the following, the different processor components with sample XML configuration are explained.

Supported processor components for the MIB Collector are as follows:

1. Delta Plugin

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifInOctets</parameter>
                        <parameter name="delta.2">ifOutOctets</parameter>
                        <parameter name="delta.3">sysUpTime.0</parameter>

                        <parameter name="result.1">deltaIfInOctets delta.1</parameter>
                        <parameter name="result.2">deltaIfOutOctets delta.2</parameter>
                        <parameter name="result.3">deltaSysUpTime delta.3</parameter>

                  </plugin>
            </processor>
      </create>
</das>
```

2. Percentage In Link Utilization

sysUpTime is measured in hundredths of a second. Hence divide this by 100.

ifSpeed is measured in bits per second. Hence multiply by 8 to get bits per second.

Step 1 Determine number of bytes sent in per second:

x = delta-InOctets / (deltaSysUpTime / 100)

Step 2 Calculate number of bits per second:

y = x * 8

Step 3 Multiply by 100% before dividing ifSpeed:

z = y * 100

Step 4 Divide by ifSpeed:

utilization = z / ifSpeed

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifInOctets</parameter>
                        <parameter name="delta.2">sysUpTime.0</parameter>
                        <parameter name="var.1">ifSpeed</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">8</parameter>
                        <parameter name="op.1">divide delta.2 int.1</parameter>

                        <parameter name="op.2">divide delta.1 op.1</parameter>
                        <parameter name="op.3">multiply op.2 int.2</parameter>
                        <parameter name="op.4">multiply op.3 int.1</parameter>
                        <parameter name="op.5">divide op.4 var.1</parameter>
                        <parameter name="result.1">pctInLinkUtilization op.5</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

3. Percentage Out Link Utilization

Step 1 Determine number of bytes sent out per second:

x = delta-OutOctets / (deltaSysUpTime / 100)

Step 2 Calculate number of bits per second:

y = x * 8

Step 3 Multiply by 100% before dividing ifSpeed:

z = y * 100

Step 4 Divide by ifSpeed:

utilization = z / ifSpeed

```
<das>
```

```
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifOutOctets</parameter>
                        <parameter name="delta.2">sysUpTime.0</parameter>
                        <parameter name="var.1">ifSpeed</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">8</parameter>
                        <parameter name="op.1">divide delta.2 int.1</parameter>

                        <parameter name="op.2">divide delta.1 op.1</parameter>
                        <parameter name="op.3">multiply op.2 int.2</parameter>
                        <parameter name="op.4">multiply op.3 int.1</parameter>
                        <parameter name="op.5">divide op.4 var.1</parameter>
                        <parameter name="result.1">pctOutLinkUtilization op.5</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

4. Percentage In Link Utilization with Configured Bandwidth

%InLinkUtilization = delta-InOctets * 100/(bandwidth.i * interval)

Here interval is delta-sysUpTime.

int.2 is the parameter to be used for user-configurable bandwidth.

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifInOctets</parameter>
                        <parameter name="delta.2">sysUpTime.0</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="int.2">512000</parameter>
                        <parameter name="op.1">multiply delta.1 int.1</parameter>
                        <parameter name="op.2">divide delta.2 int.1</parameter>
                        <parameter name="op.3">multiply op.2 int.2</parameter>
                        <parameter name="op.4">divide op.1 op.3</parameter>
                        <parameter name="result.1">pctInLinkUtilBW op.4</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

5. Percentage Out Link Utilization with Configured Bandwidth

%OutLinkUtilization = delta-OutOctets * 100/(bandwidth.i * interval)

Here interval is delta-sysUpTime.

int.2 is the parameter to be used for user configured bandwidth.

```
<das>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">ifOutOctets</parameter>
                        <parameter name="delta.2">sysUpTime.0</parameter>
                        <parameter name="int.1">100</parameter>
```

```
                        <parameter name="int.2">512000</parameter>
                        <parameter name="op.1">multiply delta.1 int.1</parameter>
                        <parameter name="op.2">divide delta.2 int.1</parameter>
                        <parameter name="op.3">multiply op.2 int.2</parameter>
                        <parameter name="op.4">divide op.1 op.3</parameter>
                        <parameter name="result.1">pctOutLinkUtilBW op.4</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

6. AAA Server plugin

Include casAuthenTransaction in front of every attribute name.

%failures = (delta-failures.i) * 100 / (delta-failures.i + delta.successes.i)

```
<das>
      <load>
            <mib name="CISCO-AAA-SERVER-MIB.my"/>
      </load>
      <create>
            <processor name="P1">
                  <plugin name="das.jar">
                        <class>GenericFormula</class>
                        <parameter name="delta.1">casAuthenTransactionFailures</parameter>
                        <parameter
name="delta.2">casAuthenTransactionSsuccesses</parameter>
                        <parameter name="int.1">100</parameter>
                        <parameter name="op.1">multiply delta.1 int.1</parameter>
                        <parameter name="op.2">plus delta.1 delta.2</parameter>
                        <parameter name="op.3">divide op.1 op.2</parameter>
                        <parameter name="result.1">pctCasAuthenFailuresBW op.3</parameter>
                  </plugin>
            </processor>
      </create>
</das>
```

Plugin class files are placed in the **das.jar** file in the **/plugin** directory of the Cisco CNS PerfE installation directory.

First load the plugin as follows:

```
<das>
      <load>
            <plugin name="das.jar"/>
      </load>
</das>
```

Configure Cisco CNS PerfE to collect the required data for computation and associate the required processor component for computing the resulting data.

Sample configuration for the Delta plugin, where “P1” is defined as one of the plugins described earlier:

```
<das>
      <create>
            <collector name="MC">
                  <MibCollector>
                        <processor name="P1"/>
                         <schedule name="MibSchedule"/>
                        <device name="MD"/>
                        <dataHandler name="MibExport"/>
                        <snmpGet>
                              <oid>sysUpTime.0</oid>
```

```
                        </snmpGet>
                        <snmpGetMany>
                              <oid>ifInOctets</oid>
                              <oid>ifOutOctets</oid>
                        </snmpGetMany>
                  </MibCollector>
            </collector>
      </create>
      <start>
            <collector name="MC"/>
      </start>
</das>
```

Formatter Plugin

The export file of the MIB Collector will contain the type of the MIB attribute in addition to its default output when configured with the Formatter plugin.

```
<das>
      <load>
            <plugin name="das.jar"/>
      </load>
      <create>
            <formatter name="F1">
                  <plugin name="das.jar">
                        <class>MibTypeFormatter</class>
                  </plugin>
            </formatter>

            <schedule name="MibSchedule">
                  <start>2002-03-06T00:00:00.000-08:00</start>
                  <interval>PT2M</interval>
                  <interTaskDelay>PT0.1S</interTaskDelay>
            </schedule>

            <device name="MD">
                  <snmp>
                        <ipaddress>10.77.21.92</ipaddress>
                        <readCommunity>public</readCommunity>
                        <writeCommunity>private</writeCommunity>
                  </snmp>
            </device>

            <dataHandler name="MibExport">
                  <formatter name="F1"/>
                        <url>
                              <prefix>ftp://guest:guest@10.77.12.94/%2Fdata/MibPeriodic</p
refix>
                        </url>
            </dataHandler>
            <collector name="MC">
                  <MibCollector>
                        <schedule name="MibSchedule"/>
                        <device name="MD"/>
                        <dataHandler name="MibExport"/>
                        <snmpGet>
                              <oid>sysUpTime.0</oid>
                        </snmpGet>
                        <snmpGetMany>
                              <oid>ifInOctets</oid>
                        </snmpGetMany>
                  </MibCollector>
```

```
            </collector>
      </create>
      <start>
            <collector name="MC"/>
      </start>
</das>
```

Data Export

The following shows the output format of the data export file (default output without Formatter plugin).

```
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifOutOctets.4,1485307
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifOutOctets.3,953335
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifOutOctets.2,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifOutOctets.1,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,sysUpTime.0,7734066
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfOutOctets.4,3326
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfOutOctets.3,1464
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaSysUpTime.0,12008
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfOutOctets.2,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfOutOctets.1,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifInOctets.4,4218748
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifInOctets.3,1421588
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifInOctets.2,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,ifInOctets.1,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfInOctets.4,6899
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfInOctets.3,1742
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfInOctets.2,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaIfInOctets.1,0
MC,10.77.11.92,2002-10-25T11:06:00.012Z,deltaSysUpTime.0,12008
```

Formatter plugin output:

```
CollectorName,DeviceAddress,Timestamp,AttributeName,AttributeValue,AttributeType
MC,10.77.11.92,2002-10-18T03:44:00.000Z,ifInOctets.16,0,Counter32
MC,10.77.11.92,2002-10-18T03:44:00.000Z,ifInOctets.15,0,Counter32
MC,10.77.11.92,2002-10-18T03:44:00.000Z,ifInOctets.14,0,Counter32
MC,10.77.11.92,2002-10-18T03:44:00.000Z,ifInOctets.12,0,Counter32
MC,10.77.11.92,2002-10-18T03:44:00.000Z,sysUpTime.0,101561860,TimeTicks
MC,10.77.11.92,2002-10-18T03:44:00.000Z,ifInOctets.9,0,Counter32
```

The plugin data exported by the MIB Collector is in the following order:

Table 6-7 Order of the Plugin Data Exported by the MIB Collector

Order	Parameter
1.	collector name
2.	device IP address
3.	data collection timestamp
4.	attribute name
5.	attribute value
6.	attribute type

Note All non-SNMP attributes that are derived by the Processor plugin and configured with the Formatter plugin appear with an "unknown" attribute type.

Threshold Processing on Plugin Data

Thresholds can be configured on the plugin data.

For example:

```
<threshold name="MibThreshold">
      <attribute name="deltaSysUpTime.0">
            <raiseOperation>GT</raiseOperation>
            <raiseValue>100</raiseValue>
            <clearOperation>LT</clearOperation>
            <clearValue>50000</clearValue>
            <level>critical</level>
      </attribute>
 </threshold>
```

Note If delta value of any attribute is involved in computation, then the first exported file will contain only raw data (i.e., it will not contain any computed data). Subsequent collections will export both raw and computed data.

Note IntComponent always has to be provided as a second operand to OpComponent. Thus, the following format is not allowed:

```
<parameter name="op.3">divide int.1 op.2</parameter>
```

Note If sysUpTime attribute is not present as part of processor plugin when some delta on some attribute needs to be calculated, then first set of delta values calculated after system reboot is not proper values. It is recommended to poll sysUpTime so that delta values can be calculated accurately even after device reboot.

NetFlowMediator Collector

This collector collects MPLS VPN usage data for billing and reporting purposes. It collects NetFlow data from NetFlow FlowCollector and aggregates the data for periods up to one hour. The NetFlowMediator Collector also maps the network flow data with VPN customer information via its VPNSC integration module.

Requirements

Only one instance of NetFlowMediator Collector may be configured per Cisco CNS PerfE system.

Correct use of this collector also requires proper configuration of VPN Solution Center, NetFlow FlowCollector, and the NetFlow agent in IOS.

XML Configuration

The following XML tags can be configured for a NetFlowMediator Collector:

Table 6-8 Tags in the XML Configuration for the NetFlowMediator Collector

XML Tag	Purpose	Mandatory/ Optional
aggregatePeriod	Aggregation period in minutes. The value must be between 1 and 60 minutes.	Mandatory
aggregateDataFormat	Aggregated data format: use **bin** or **ascii**.	Mandatory
VPNSC	VPNSC host machine. If Cisco CNS PerfE name has more than two dots, use IP address of the VPNSC host.	Mandatory
VPNSCUser	VPNSC administrator name.	Mandatory
VPNSCPasswd	VPNSC administrator password.	Mandatory

The following is a sample XML configuration for NetFlowMediator Collector. Note that the schedule ("Schedule"), the devices ("dev1" and "dev2"), the notifiers ("ntf1" and "ntf2"), and the purger ("purger"), must be defined prior to defining the collector.

```
<collector name="nfc1">
      <NetFlowCollector>
            <schedule name="Schedule"/>
            <device name="dev1"/>
            <device name="dev2"/>
            <notifier name="ntf1"/>
            <notifier name="ntf2"/>
            <purger name="purger"/>
            <aggregatePeriod>60</aggregatePeriod>
            <aggregateDataFormat>bin</aggregateDataFormat>
            <VPNSC>alexrtp-u10</VPNSC>
            <VPNSCUser>admin</VPNSCUser>
            <VPNSCPasswd>admin</VPNSCPasswd>
      </NetFlowCollector>
</collector>
```

Data Export

The output from the NetFlowMediator Collector is done on an hourly basis. The output is in a NetFlow FlowCollector file format in either Binary or ASCII format. Each time an output file is created, the **filesready** file is updated to indicate which files have been produced. These output files can be retrieved via FTP.

The output data files are stored using in following path and file name:

<CNS_PerfE_Home>*/data/nfc/outputfiles/*<NFC host name>*/*<RouterIP>*/*Date*/MPLSVPNUsage/<RouterIP>.*TimeStamp

where

- *NFC host name* is the host name or IP address of the NetFlow FlowCollector, from which the data is collected
- *RouterIP* is the IP address of the router
- *Date* is the date on which the data was exported

- *TimeStamp* is the time that the data was exported

When data is exported in ASCII format the items in the data header are as follows:

- Source–The Router IP.
- Format 2.
- Aggregation MPLSVPNUsage.
- Period–The number of minutes in the period, up to 60. If the period is equal to 0, then the contents of the file is a partial result. Partial results are generated when the collector is shut down but there are partial aggregation results in memory.
- Start time–Start time of NetFlowCollector record.
- Endtime–EndTime of NetFlowCollector record.
- Flow–The number of flows.
- Missed–The number of missed records.
- Records–The number of records.

The data that is exported in each row of the file is described in the following table. If this data is exported in ASCII format, the fields are separated by a "|".

Table 6-9 Order of the Data Exported by NetFlowMediator Collector (ASCII Format)

Order	Parameter
1.	Destination Address
2.	Input
3.	TOS
4.	Packets
5.	Octets
6.	Flows
7.	Customer
8.	Site
9.	CE

You can export the data in binary format. The binary format is defined below:

```
typedef struct {
    /* Keys */
    u_long srcaddr;            /* Source IP */
    u_long dstaddr;            /* Destination IP */
    u_short input;  /* interface */
    u_char tos;                /* Type of Service */
    char pad1;                  /* Data alignment */

    /* Values */
    u_long pkts;               /* Packet count */
    u_long octets;             /* Byte count */
    u_long flows;              /* Flow count */
    u_short custlength;        /* total bytes of following customer's data */

    u_short pad2;
    char[] customer;  /* variable length string */
    char[] site;  /* variable length string */
```

```
    char[] ce;    /* variable length string */
};
```

NFC-like **filesready** files are generated daily in ***<CNS_PerfE_Home>*/data/nfc/logs** directory (**filesready.<*date*>**). This file contains the file names of output data files.

Threshold Crossing Alerts and Hotspot Polling

This collector does not support threshold crossing alerts or hotspot polling.

Purging Data

It is recommend that the NetFlowMediator Collector purge be run at minimum on the following schedule. Disk space usage may require that data be purged more frequently. The following purge schedule purges data every 24 hours. Each time data is purged, any data that is older than 48 hours is purged.

```
<purger name="purger">
      <time>2001-12-01T00:00:00.000Z</time>
      <interval>PT24H</interval>
      <delay>PT48H</delay>
</purger>
```

Limitations

Using the NetFlowMediator Collector in conjunction with any other Cisco CNS PerfE collector (other than the System Collector) is *not* supported; it must be the only Cisco CNS PerfE collector running.

Usage requires proper configuration of the following software packages:

- Cisco VPN Solution Center v 2.0.0.9 or later
- Cisco NetFlow FlowCollector v 3.6 or later
- Cisco IOS NetFlow Agent

This collector is valid only for capturing and reporting on MPLS VPN traffic on CNS PerfE ingress interface. There is a limitation of one VPN per PE (sub)interface.

MPLS/VPN PE-PE Traffic Reports

In Cisco CNS PerfE, Release 2.1, the NetFlow Mediator Collector can alternatively be used to produce MPLS/VPN PE-PE traffic reports. All Provider Edge (PE) routers in a provider network export traffic statistics (NetFlow Data Export or NDE) to a few hosts that run NetFlow Collection Engine (NFC). These NFCs conduct aggregation of the NDE they receive. The NetFlow Mediator then collects aggregation results from the NFCs and produces the reports.

Note This feature requires CNS NetFlow Collection Engine, release 5.0.

Configuration

Provider Edge (PE) Routers

On PEs, NetFlow should be enabled on all CE-facing interfaces with the following command:

```
Router(config-if)# ip route-cache flow
```

BGP next hop field is required by this feature. To enable BGP next hop export (supported since IOS 12.0(26)S), use the following commands at the global level:

```
Router# ip flow-export version 9 bgp-nexthop
Router# ip flow-export destination <NFC host IP address> <NFC port number>
```

NetFlow Collection Engine 5.0

On the NetFlow Collection Engine, create an aggregator with BGP next hop as an aggregation key in **<NFC Base Directory>/config/nfc-config.xml**. Follow these steps:

Step 1 Add two more value builders in **<value-builders>** if they are not defined:

```
<sum-value id="packet-count-32-value">
<field>PKTS_32</field>
</sum-value>

<sum-value id="byte-count-32-value">
<field>BYTES_32</field>
</sum-value>
```

Step 2 Add one more key builder in **<key-builders>** for the BGP next hop:

```
<inetaddress-key id="bgp-next-hop-key">
<field>BGP_NEXT_HOP</field>
</inetaddress-key>
```

Step 3 Add one new aggregation scheme in **<aggregation-schemes>**:

```
<aggregation-scheme id="nexthop">
<keys>
<key id="bgp-next-hop-key"/>
</keys>
<values>
<value id="packet-count-32-value"/>
<value id="byte-count-32-value"/>
</values>
</aggregation-scheme>
```

Step 4 Add the aggregator in **<aggregators>**:

```
<aggregator id="bgpnexthop">
<aggregation-scheme id="nexthop"/>
<period-minutes>15</period-minutes>
<port protocol="udp">8888</port>
<writer>
<ascii-writer>
<use-compression>false</use-compression>
<filename-includes-date>false</filename-includes-date>
</ascii-writer>
</writer>
</aggregator>
```

CNS PerfE

The following is a sample XML configuration relevant to PE-PE traffic reports:

```
<?xml version="1.0" encoding="UTF-8"?>
<das>
    <create>
        <schedule name="Schedule">
            <start>2001-10-25T00:00:00.000-08:00</start>
            <interval>PT15M</interval>
        </schedule>
        <device name="dev1">
            <ftp>
                <ipaddress>nfchost1</ipaddress>
                <username>nfcuser</username>
                <password>nfcuser</password>
            </ftp>
        </device>
        <device name="dev2">
            <ftp>
                <ipaddress>nfchost2</ipaddress>
                <username>nfcuser</username>
                <password>nfcuser</password>
            </ftp>
        </device>
        <threshold name="th1">
            <attribute name="sys.disk.home.usedPercentage">
          <raiseOperation>GT</raiseOperation>
                <raiseValue>80</raiseValue>
                <clearOperation>LT</clearOperation>
                <clearValue>70</clearValue>
                <level>minor</level>
            </attribute>
        </threshold>
        <notifier name="ntf1">
            <cns>
                <subject>cisco.mgmt.das.notifier-listener</subject>
            </cns>
        </notifier>
        <notifier name="ntf2">
            <trap>
                <ipaddress>scmrtp2-u30</ipaddress>
            </trap>
        </notifier>
        <purger name="purger">
            <time>2001-12-01T00:00:00.000Z</time>
            <interval>PT24H</interval>
            <delay>PT48H</delay>
        </purger>
        <collector name="nfc1">
            <NetFlowCollector>
         <schedule name="Schedule"/>
            <device name="dev1"/>
            <device name="dev2"/>
               <notifier name="ntf1"/>
           <notifier name="ntf2"/>
             <purger name="purger"/>
             <reports>
          <pe-pe-traffic-matrix>
           <pe-list-file>/tmp/cnsperfe/config/peList</pe-list-file>
           <nfc-agg-scheme>nexthop</nfc-agg-scheme>
          </pe-pe-traffic-matrix>
          <reports-time-level>daily</reports-time-level>
        </reports>
```

```
            </NetFlowCollector>
        </collector>
        <collector name="sys">
            <schedule name="Schedule"/>
         <threshold name="th1"/>
            <notifier name="ntf1"/>
         <notifier name="ntf2"/>
            <SystemCollector/>
        </collector>
    </create>
    <start>
        <collector name="nfc1"/>
        <collector name="sys"/>
      </start>
      </das>
```

The **<pe-list-file>** gives the location of a file which contains IP addresses of all PEs in the provider network. Entries in NFC results that do not match any address in the list will not be included in the PE-PE traffic reports. If this file is not given here or contains no IP addresses in it, no filtering will be done and all entries in NFC results will be reported. The contents of a sample PE list file are as follows (assuming 1.1.1.1, 2.2.2.2 and 3.3.3.3 are all of the PEs in the provider network):

```
1.1.1.1
2.2.2.2
3.3.3.3
```

Also note that the **<nfc-agg-scheme>** specifies results of which aggregation scheme will be fetched from the NFC hosts. The value given here should exactly match the aggregation scheme ID specified in the NFC configuration.

Output Format

The PE-PE traffic reports will be in CSV format. A sample report looks as follows:

```
Daily PE-PE Traffic
Start: 2003-01-31T00:00:00.001
End: 2003-02-01T00:00:00.000
Ingress\Egress,1.1.1.1,2.2.2.2,3.3.3.3
1.1.1.1,N/A,100334/135343,150343/200342
2.2.2.2,200234/250343,N/A,300344/400343
3.3.3.3,400432/500343,250645/350343,N/A
```

The report has header and a data matrix. The first three rows here are header. It specifies the report frequency and type and also contains start/end time stamps. Each data row reflects traffic from a certain PE, whose IP address is displayed in the first cell of the row; each data column reports traffic to a certain destination PE. The numbers displayed in cells are packet count and K-byte count, separated with a slash. For example, the 400432/500343 pair in the bottom row shows that there are 400432 packets containing 500343 K-bytes routed from PE 3.3.3.3 to PE 1.1.1.1, on 1/31/2003. N/A stands for not applicable.

All reports will be stored under the **$DAS_HOME/data/nfc** directory with the following file structure:

```
$DAS_HOME/data/nfc/reports/<report-time-level>/<report-type>/<report-type>.timestamp
```

The report time level shows how frequently reports are. Supported frequencies are Hourly and Daily. The report type is PE-PE-traffic-matrix. The time stamp will be in the following format: yyyy_mm_dd.hhmm. For instance, 2003_02_01.0000.

Reports will be kept for a certain length of time, according to the <purger> configuration as shown in the sample XML.

RADIUS Collector

The RADIUS Collector receives RADIUS accounting requests, and returns an accounting response (an acknowledgement) for each request back to the gateway. Cisco CNS PerfE acts as a RADIUS accounting server and the gateway acts as the client. Transactions between gateways and the RADIUS accounting server are authenticated through the use of a shared secret which is never sent over the network. A RADIUS accounting packet is encapsulated in a UDP datagram. The default UDP port on Cisco CNS PerfE as the RADIUS accounting server is 1813. The UDP port is also configurable through the XML interface.

A Cisco VoIP call may consist of several call legs, where each leg represents a part of the path of a complete call. Cisco CNS PerfE can receive **start** and **stop** accounting requests but it will only process **stop** requests which are sent when a call for a call leg is completed.

Each call processed through a gateway consists of an incoming and an outgoing call leg. Call legs 1 and 2 represent incoming and outgoing calls for the originating gateway while call legs 3 and 4 represent incoming and outgoing for the terminating gateway of a four leg call. This information is summarized in Table 6-10.

Table 6-10 Call Leg Description

Call Leg Name	Call Leg Type	Description
Answer/Telephony	1	This call leg originates from POTS network into the gateway. The gateway “answers” the call from the POTS device.
Originate/VoIP	2	This call leg originates from the gateway into the IP Network.
Answer/VoIP	3	This call leg originates from the IP network into the gateway. The gateway “answers” the VoIP call.
Originate/Telephony	4	This call leg originates from the gateway into the POTS network.

The RADIUS Collector correlates the RADIUS stop data, Call History data (if a CallHistory Collector is configured), and SS7 call legs from BAMS (if a BAMS Collector is configured). Only one RADIUS Collector, one BAMS Collector, and one CallHistory Collector can be configured on a single Cisco CNS PerfE system. By default, the correlation is performed based on the schedule assigned to the RADIUS Collector. If the CallHistory Collector or BAMS Collector is configured, the recommended value for Schedule Interval should be at least twice the value of the collection interval for CallHistory or BAMS, whichever is greatest.

The RADIUS Collector settings are specified in the **das.properties** file located at **<*CNS_PerfE_Home*>/config/das.properties** and include the following properties:

Table 6-11 RADIUS Collector Settings in das.properties File

Property Name	Purpose
radius.req.threads=4	Number of request processing threads.
radius.secret=cisco	Key to generate the authenticator. Deprecated setting. Recommend using XML property.

For an overview of the file format used by the RADIUS Collector, see Appendix A, "RADIUS Collector Export Data File Format".

For more detailed information, see the "For More Information" section on page 1-10.

Requirements

AAA broadcast accounting in Cisco Voice Gateways enables AAA accounting records to be transmitted to both a Cisco CNS PerfE and a RADIUS AAA server. Voice gateways should be configured so that only stop records are sent to Radius Collector since Cisco CNS PerfE only responds to stops. All gateways sending accounting requests to Cisco CNS PerfE should be configured with the same RADIUS secret.

XML Configuration

The RADIUS Collector has three attributes that you can configure through the XML interface:

Table 6-12 Tags in the XML Configuration for the RADIUS Collector

XML Tag	Purpose	Mandatory/ Optional
acctPort	The UDP destination port for the RADIUS server to listen to for Radius packets. The schedule specified in the collector can specify a start time, but the interval from the schedule is not used.	Mandatory
ageInterval	This attribute determines when to write the call legs to the RADIUS file in the **<*CNS_PerfE_Home*>/data/radius** directory if all four of the legs have not been received. This value should be less than the **fileInterval** value. The default for ageInterval is 30 seconds.	Optional
fileInterval	This attribute determines when to generate the RADIUS file. The default for **fileInterval** is half of the **scheduleInterval**.	Optional
radiusSecret	This attribute defines the RADIUS secret key value. If this parameter is not provided in the configuration, then the value from **das.properties** will be used.	Optional

Sample XML configuration:

```
<collector name="radiusCollector">
```

```
      <radiusCollector>
            <schedule name="RADIUSSCHEDULE"/>
            <acctPort>1813</acctPort>
            <ageInterval>PT15S</ageInterval>
            <fileInterval>PT5M</fileInterval>
            <radiusSecret>cisco</radiusSecret>
      </radiusCollector>
</collector>
```

Schedule Interval is an attribute specified in the schedule**.** It determines how often to generate the VoipCorrelator files. The value should be greater than the **fileInterval** value.

Sample XML configuration:

```
<collector name="radiusCollector">
      <radiusCollector>
            <schedule name="RADIUSSCHEDULE"/>
            <acctPort>1813</acctPort>
            <ageInterval>PT15S</ageInterval>
            <fileInterval>PT5M</fileInterval>
      </radiusCollector>
</collector>
```

Suppose that **ageInterval** is set to 30 seconds, **fileInterval** is 5 minutes, and Schedule (correlation) Interval is 15 minutes.

In this case, Cisco CNS PerfE starts aging out radius stop packets at 30 seconds and flushes the CDRs to data files in the **/radius** directory. If **fileInterval** is 5 minutes, the Radius Collector generates an intermediate file every 5 minutes. With a schedule interval of 15 minutes, the VoIPCorrelator correlates the radius/callhistory/BAMS CDRs every 15 minutes.

Data Export

In Cisco CNS PerfE, the RADIUS Collector collects RADIUS accounting packets, correlates RADIUS Call Detail Records (CDR), and the correlated RADIUS CDRs are written to a data file. The data is generated in Comma Separated Values (CSV) format. The name of this file is passed to the VoipCorrelator module. The VoipCorrelator module correlates RADIUS, Call History, and BAMS call data and saves the results in flat files in the ***<CNS_PerfE_Home>*/data/VoipCorrelator** directory. The PM application should retrieve files from the ***<CNS_PerfE_Home>*/data/VoipCorrelator** directory and it should delete the files.

The filenames are <*dasname*>_<*timestamp*>.

For example:

nfc-ultra10_D20011219T193632Z

nfc-ultra10_D20011219T193732Z

The files in ***<CNS_PerfE_Home>*/data/radius** and ***<CNS_PerfE_Home>*/data/callhistory** are for internal use by the system. Letting an external application use these files will cause unexpected behavior.

Note The RADIUS Collector logs CDR Statistics to the ***<CNS_PerfE_Home>*/logs/cdr.log** file. This file identifies how many CDRs are written to each data file.

The XML elements for the correlated data are as follows:

```
<calls>
      <call id="ABCD1234 ABCD1234 ABCD1234 ABCD1234">
```

```
            <callLegCount>4</callLegCount>
            <callLeg>...</callLeg>
            <callLeg>...</callLeg>
            <callLeg>...</callLeg>
            <callLeg>...</callLeg>
            <bams>...</bams>
      </call>
</calls>
```

Note The **call id** field represents the **h323-conf-id**. The schema for the data files is in ***<CNS_PerfE_Home>*/schema/calls.xsd**.

Below is an example of a RADIUS data file:

```
<calls>
    <call id="CBB9FD53 8D613DAF 0 847B65B8">
        <callLegCount>2</callLegCount>
        <callLeg>VoIP,answer,001B7B14,0,01068186663,1003148814335,bj-3t-rs1-
        gw3.das.com.,211.93.196.232,1003148814335,1,21:26:27.663 Beijing Tue Mar 5
        2002,21:26:50.591 Beijing Tue Mar 5 2002,21:26:50.591 Beijing Tue Mar 5
        2002,10,0,211.101.116.45,0,76,0,4,0,CBB9FD53 8D613DAF 0 847B65B8,
        subscriber=Unknown|pre-bytes-in=0|pre-bytes-out=0|pre-paks-in=0|pre-paks-
        out=0|nas-rx-speed=0|nas-tx-speed=0</callLeg>
        <callLeg>Telephony,originate,001B7B15,0,01068186663,1003148814335,bj-3t-rs1-
        gw3.das.com,,1003148814335,1,21:26:27.707 Beijing Tue Mar 5 2002,21:26:50.623
        Beijing Tue Mar 5 2002,21:26:50.623 Beijing Tue Mar 5
        2002,10,0,211.101.116.45,76,0,4,0,0,CBB9FD53 8D613DAF 0 847B65B8,
        subscriber=Unknown|h323-ivr-out=Tariff:Unknown|pre-bytes-in=0|pre-bytes-out=0|pre-
        paks-
        in=0|pre-paks-out=0|nas-rx-speed=0|nas-tx-speed=0</callLeg>
    </call>
    <call id="47D21F4A 2F7311D6 8AA48479 4B9315DF">
        <callLegCount>2</callLegCount>
        <callLeg>Telephony,answer,0000F516,69,02164104400,1002223061396,tj-c-
        gw1.das.com,,131002908461,1,21:25:22.670 CST Tue Mar 5 2002,21:25:22.691 CST Tue
        Mar 5 2002,21:26:31.491 CST Tue Mar 5
        2002,10,0,211.101.120.10,3572,160440,133,1286,0,47D21F4A 2F7311D6 8AA48479
        4B9315DF, subscriber=RegularLine|tariff-type=Unknown|pre-bytes-in=0|pre-bytes-
        out=0|pre-paks-in=0|pre-paks-out=0|connect-progress=101|nas-rx-speed=0|nas-tx-spee
        d=0</callLeg>
        <callLeg>VoIP,originate,0000F5AF,13,02164104400,1002223061396,tj-c-
        gw1.das.com,211.101.117.53,131002908461,1,21:26:08.574 CST Tue Mar 5
        2002,21:26:18.827 CST Tue Mar 5 2002,21:26:31.507 CST Tue Mar 5
        2002,10,0,211.101.120.10,11360,1978,355,69,10,47D21F4A 2F7311D6 8AA48479 4B9315DF,
        subscriber=RegularLine|pre-bytes-in=0|pre-bytes-out=0|pre-paks-
        in=0|pre-paks-out=0|connect-progress=101|nas-rx-speed=0|nas-tx-speed=0</callLeg>
    </call>
    <call id="CD43276A 2F7311D6 8F648479 4B9315DF">
        <callLegCount>1</callLegCount>
        <callLeg>Telephony,answer,0000F76B,9,00000,1002226030493,tj-c-gw1.das.com,,1002226
        030493,1,21:29:06.548 CST Tue Mar 5 2002,21:29:06.570 CST Tue Mar 5
        2002,21:29:15.180 CST Tue Mar 5 2002,10,0,211.101.120.10,0,25440,0,159,0,CD43276A
        2F7311D6 8F648479
        4B9315DF,subscriber=RegularLine|tariff-type=Unknown|pre-bytes-in=0|pre-
        bytes-out=0|pre-paks-in=0|pre-paks-out=0|connect-progress=101|nas-rx-speed=0|nas-
        tx-speed=0</callLeg>
    </call>
</calls>
```

Order of Call Leg Data

The data for each call leg is listed in the CSV data file in the following order:

Table 6-13 Order of the Data Stored for Each Call Leg

Order	Parameter Name
1.	callType
2.	callOrigin
3.	sessionId
4.	sessionTime
5.	calledStationId
6.	callingStationId
7.	gatewayId
8.	remoteAddress
9.	userName
10.	serviceType
11.	setupTime
12.	connectTime
13.	disconnectTime
14.	disconnectCause
15.	nasPortType
16.	nasIpAddress
17.	inputOctets
18.	outputOctets
19.	inputPackets
20.	outputPackets
21.	voiceQuality
22.	incomingConfId
23.	vsa name-value pairs

Order of BAMS Data

The BAMS data is listed in the CSV data file in the following order:

Table 6-14 Order of the Data Stored for BAMS

Order	Parameter Name
1.	cdb_identifier
2.	cdb_version
3.	cdb_timepoint
4.	call_reference_id

Table 6-14 Order of the Data Stored for BAMS (continued)

Order	Parameter Name
5.	iam_timepoint
6.	acm_timepoint
7.	anm_timepoint
8.	originating_trunk_group
9.	originating_member
10.	calling_number
11.	charged_number
12.	dialed_number
13.	called_number
14.	terminating_trunk_group
15.	terminating_member
16.	first_release_source
17.	vsc_info_field
18.	iam_timepoint_rcvd_ms
19.	iam_timepoint_sent_ms
20.	acm_timepoint_rcvd_ms
21.	acm_timepoint_sent_ms
22.	anm_timepoint_rcvd_ms
23.	anm_timepoint_sent_ms
24.	first_rel_timepoint_ms
25.	second_rel_timepoint_ms
26.	rlc_timepoint_rcvd_ms
27.	rlc_timepoint_sent_ms
28.	ansi_calling_party_category
29.	ansi_user_service_information
30.	ansi_calling_number_nature_of_address
31.	ansi_charged_number_nature_of_address
32.	ansi_dialed_number_nature_of_address
33.	ansi_called_number_nature_of_address
34.	ansi_reason_code
35.	ansi_transit_network_selection
36.	ansi_carrier_selection_parameter
37.	itu_calling_party_category
38.	itu_user_service_information
39.	itu_calling_number_nature_of_address
40.	itu_charged_number_nature_of_address

Table 6-14 Order of the Data Stored for BAMS (continued)

Order	Parameter Name
41.	itu_dialed_number_nature_of_address
42.	itu_called_number_nature_of_address
43.	itu_reason_code
44.	vsc_id
45.	subscriber_duration
46.	network_usage_duration
47.	redirecting_number

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

Cisco CNS PerfE purges the correlated files based on the purge parameters specified in **das.properties**. However, a purger can be configured for RADIUS Collector. In the usual operations, the PM or the NMS application should delete files after the files are retrieved.

Hotspot polling, threshold processing, on-demand data export using XML interface, and periodic scheduling of data export are not supported for this collector.

Limitations

In order for VoipCorrelation to work correctly, the RADIUS Collector must be started before Call History or BAMS collectors.

The RADIUS Collector does not support RADIUS attribute 44, **Acct-Session-Id overloading**. You must enable Vendor-Specific Attributes (VSAs), by using the **gw-accounting h323 vsa** command on voice gateways. See Appendix A, "RADIUS Collector Export Data File Format," for a list of VSAs.

SAA Collectors

SAA Collectors collect performance data measured by the Service Assurance Agent (SAA) within an IOS device. They provide different types of measurements by simulating different types of protocols. SAA is used to measure performance statistics between two routers. The source router is where SAA operations are configured. The target router is usually configured with the SAA responder.

SAA collectors interact with a Cisco device that has an IOS image, which supports SAA operations (not all IOS images support SAA). The collector uses SNMP to control and retrieve response time information from the Cisco device. The collected data is retrieved from "rtr probe", which measures data sent between a source Cisco device to a destination Cisco device. SAA collectors in general require "rtr responder" to be enabled at the target device via a command line interface using "configure terminal" at the Cisco device.

SAA Collectors in Cisco CNS PerfE

There are three SAA collectors in Cisco CNS PerfE:

- SAAIcmpEcho Collector, page 6-79
- SAAJitter Collector, page 6-81
- SAAUdpEcho Collector, page 6-87

For all SAA Collectors, refer to references on SAAgent:

- Configuration Guide– http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/ffun_c/fcfprt3/fcf017.htm
- Command Reference– http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/ffun_r/ffrprt3/frf017.htm
- Product Feature Guide– http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a00801d3a94.html#1060419

Configuration of the SAA Operation

If the SAA <operation> tag is not used, a default operation will be created for backward compatibility. The preferred way is to configure the <operation> tag.

Parameters configured outside of the <operation> tag will be used as default parameters for the operation. Thus, if a parameter is missing within the <operation> tag, it will look for the parameter outside the tag.

If index is provided:

Before the collector is started, the operation with the index must be configured and running on the router and the rtr responder, if required, must be configured also.

Other parameters specified in the XML will be ignored.

If index is not provided:

Before the collector is started, the rtr responder has to be configured for the SaaJitter or SaaUdpEcho collectors.

Parameters specified in the XML will be used by Cisco CNS PerfE to automatically create the operation on the source router.

SAAIcmpEcho Collector

The SAAIcmpEcho Collector measures the round trip time for ICMP between source and destination devices.

Requirements

None.

XML Configuration

The following table describes the attributes that you can set for the SAAIcmpEcho Collector.

Table 6-15 Tags in the XML Configuration for the SAAIcmpEcho Collector

XML Tag	Descriptions	Mandatory/ Optional
index	The index to an existing operation already created on the router.	Optional
packetSize	The payload size of the ICMP ping packet. The default value is 64.	Optional
targetAddress	The IP address of the device on which the SAA operation occurs. The default target address is the one specified outside the <operation> tag.	Mandatory
timeout	The amount of time to wait for a response to the operation. The default value is 5 seconds.	Optional
VrfName	The vrfName in which the saa operation should use.	Optional

Sample XML configuration:

```
<collector name="SAAicmp1">
      <SaaIcmpEchoCollector>
            <schedule name="S1"/>
            <device name="dev1">
                  <operation name="op1">
                        <targetAddress>10.0.0.1</targetAddress>
                  </operation>
                  <operation name="op2">
                        <targetAddress>10.0.0.2</targetAddress>
                  </operation>
            </device>
            <dataHandler name="dh1"/>
      </SaaIcmpEchoCollector>
</collector>
```

Data Export

The SaaIcmpEcho Collector retrieves measurements about the IcmpEcho operation and stores the data in the database. It exports data in CSV format. Data can be exported at specified frequency or on-demand using the XML interface (<get> <data> tags). The first line of each data file contains the column headings separated by commas. Each subsequent line contains the following information:

Table 6-16 Order of the Data Exported by the SaaIcmpEcho Collector

Order	Parameter
1.	collector name
2.	device name
3.	data collection timestamp
4.	rttMonLatestRttOperCompletionTime

Table 6-16 Order of the Data Exported by the SaaIcmpEcho Collector (continued)

Order	Parameter
5.	rttMonLatestRttOperSense
6.	rttMonLatestRttOperApplSpecificSense
7.	ttMonLatestRttOperSenseDescription
8.	rttMonLatestRttOperTime

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The SaaIcmpEcho Collector supports threshold crossing/clearing alerts (TCAs). TCAs can be configured only for the attribute **rttMonLatestRttOperCompletionTime**. The collector checks for TCAs after each data collection.

The SaaIcmpEcho Collector supports purging of collected performance data. This is done on a schedule configured for all of Cisco CNS PerfE, or you can configure a purger just for this collector.

The SaaIcmpEcho Collector supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not store the data internally and does not post-process the data after collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

Limitations

None.

SAAJitter Collector

The SAAJitter Collector measures the round trip time and jitter between source and destination devices. Unlike other SAA collectors, SAAJitter requires two iterations of measurements since the jitter measurement requires data from two measurements. Therefore, the first measurement would not contain values for jitter.

Requirements

A value for **rtr responder** must be configured on the target router.

This collector only works with Cisco devices, which support Service Assurance Agent.

The SaaJitter Collector is enhanced in CNS PerfE, release 2.1 to support collection of voice quality viz, rttMonLatestJitterOperMOS, and rttMonLatestOperCPIF. To support collection of these attributes, codec types and its associated parameters must e specified for operation. The following codecType values are supported:

- g711ulaw
- g711alaw
- g729a

XML Configuration

The following table describes the attributes that you can set for the SAAJitterEcho Collector.

Table 6-17 Tags in the XML Configuration for the SAAJitter Collector

XML Tag	Purpose	Mandatory/ Optional
codecNumPkts :	Number of packets to be transmitted <1-60000>	Optional
codecInterval	Inter packet interval <1-60000>	Optional
codecPacketSize	Number of bytes in payload <16-1500>	Optional
Advantage factor < 0-20> :		Optional
control	Sends a control message to the RTR responder first before the operation. The default is “enabled”.	Optional
index	The index to an existing operation already created on the router.	Optional
interval	The delay in milliseconds between each packet for an operation. The default is 20.	Optional
numOfPackets	The number of packets to send out for an operation. The default is 10.	Optional
packetSize	The payload size of the UDP packet. The default is 64.	Optional
sourcePort	The source UDP port to use. The default is 0 which means that the router will select a random port.	Optional
targetAddress	The IP address of the device on which the SAA operation occurs. The default target address is the one specified outside the <operation> tag	Mandatory
targetPort	The destination UDP port to use. The default target port is the one specified outside the <operation> tag	Mandatory
timeout	The amount of time to wait for a response to the operation.	Optional
tos	The TOS field of the IP packet. The default is 0.	Optional
VrfName	The vrfName in which the saa operation should use.	Optional

Sample XML configuration:

```
<collector name="SAAjitter">
<SaaJitterCollector>
<schedule name="S1"/>
<device name="dev1">
<operation name="op1">
<targetAddress>10.0.9.100</targetAddress>
<targetPort>101</targetPort>
</operation>
<operation name="op2">
<targetAddress>10.0.9.100</targetAddress>
<targetPort>101</targetPort>
<sourcePort>100</sourcePort>
</operation>
</device>
<timeout>PT4S</timeout>
<packetSize>256</packetSize>
<numOfPackets>15</numOfPackets>
<interval>10</interval>
<tos>0</tos>
<control>enable</control>
</SaaJitterCollector>
</collector>
```

Following is the XML configured for codec data collection

```
<collector name="SAAjitter">
<SaaJitterCollector>
<schedule name="S1"/>
<device name="dev1">
<operation name="op1">
<targetAddress>10.0.9.100</targetAddress>
<targetPort>101</targetPort>
</operation>
<operation name="op2">
<targetAddress>10.0.9.100</targetAddress>
<targetPort>101</targetPort>
<sourcePort>100</sourcePort>
</operation>
</device>
<codecType>g711alaw</codecType>
<codecNumPkts>40</codecNumPkts>
<codecInterval>100</codecInterval>
<codecPacketSize>80</codecPacketSize>
<advantageFactor>10</advantageFactor>
</SaaJitterCollector>
</collector>
```

Data Export

The SaaJitter Collector retrieves measurements about Jitter operations and stores the data in the database. It exports the data in Comma Separated Values (CSV) format. Data can be exported at specified frequency or on-demand using the XML interface (<get> <data> tags).

The first line of each data file contains the column headings separated by commas. Each subsequent line is as follows:

Table 6-18 Order of the Data Stored for the SaaJitter Collector

Order	Parameter
1.	collectorName
2.	deviceName
3.	collectionTime
4.	rttMonLatestRttOperCompletionTime
5.	rttMonLatestRttOperSense
6.	rttMonLatestRttOperApplSpecificSense
7.	rttMonLatestRttOperSenseDescription
8.	rttMonLatestRttOperTime
9.	rttMonLatestJitterOperNumOfRTT
10.	rttMonLatestJitterOperRTTSum
11.	rttMonLatestJitterOperRTTSum2
12.	rttMonLatestJitterOperRTTMin
13.	rttMonLatestJitterOperRTTMax
14.	rttMonLatestJitterOperMinOfPositivesSD
15.	rttMonLatestJitterOperMaxOfPositivesSD
16.	rttMonLatestJitterOperNumOfPositivesSD
17.	rttMonLatestJitterOperSumOfPositivesSD
18.	rttMonLatestJitterOperSum2PositivesSD
19.	rttMonLatestJitterOperMinOfNegativesSD
20.	rttMonLatestJitterOperMaxOfNegativesSD
21.	rttMonLatestJitterOperNumOfNegativesSD
22.	rttMonLatestJitterOperSumOfNegativesSD
23.	rttMonLatestJitterOperSum2NegativesSD
24.	rttMonLatestJitterOperMinOfPositivesDS
25.	rttMonLatestJitterOperMaxOfPositivesDS
26.	rttMonLatestJitterOperNumOfPositivesDS
27.	rttMonLatestJitterOperSumOfPositivesDS
28.	rttMonLatestJitterOperSum2PositivesDS
29.	rttMonLatestJitterOperMinOfNegativesDS
30.	rttMonLatestJitterOperMaxOfNegativesDS
31.	rttMonLatestJitterOperNumOfNegativesDS
32.	rttMonLatestJitterOperSumOfNegativesDS
33.	rttMonLatestJitterOperSum2NegativesDS
34.	rttMonLatestJitterOperPacketLossSD

Table 6-18 Order of the Data Stored for the SaaJitter Collector (continued)

Order	Parameter
35.	rttMonLatestJitterOperPacketLossDS
36.	rttMonLatestJitterOperPacketOutOfSequence
37.	rttMonLatestJitterOperPacketMIA
38.	rttMonLatestJitterOperPacketLateArrival
39.	rttMonLatestJitterOperSense
40.	rttMonLatestJitterErrorSenseDescription
41.	rttMonLatestJitterOperOWSumSD
42.	rttMonLatestJitterOperOWSum2SD
43.	rttMonLatestJitterOperOWMinSD
44.	rttMonLatestJitterOperOWMaxSD
45.	rttMonLatestJitterOperOWSumDS
46.	rttMonLatestJitterOperOWSum2DS
47.	rttMonLatestJitterOperOWMinDS
48.	rttMonLatestJitterOperOWMaxDS
49.	rttMonLatestJitterOperNumOfOW
50.	rttMonLatestJitterOperMOS
51.	rttMonLatestJitterOperICPIF

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The SaaJitter Collector supports threshold crossing/clearing alerts (TCAs). TCAs can be configured only for the attribute **rttMonLatestRttOperCompletionTime**. The collector checks for TCAs after each collection period. SaaJitter Collector can monitor thresholds for the following attributes:

Table 6-19 Attributes Monitored for Threshold Violations for SaaJitterColletor

Attribute
rttMonLatestRttOperCompletionTime
rttMonLatestJitterOperRTTSum
rttMonLatestJitterOperRTTSum2
rttMonLatestJitterOperRTTMin
rttMonLatestJitterOperRTTMax
rttMonLatestJitterOperMinOfPositivesSD
rttMonLatestJitterOperMaxOfPositivesSD
rttMonLatestJitterOperNumOfPositivesSD
rttMonLatestJitterOperSumOfPositivesSD
rttMonLatestJitterOperSum2PositivesSD
rttMonLatestJitterOperMinOfNegativesSD

Table 6-19 Attributes Monitored for Threshold Violations for SaaJitterColletor (continued)

Attribute
rttMonLatestJitterOperMaxOfNegativesSD
rttMonLatestJitterOperNumOfNegativesSD
rttMonLatestJitterOperSumOfNegativesSD
rttMonLatestJitterOperSum2NegativesSD
rttMonLatestJitterOperMinOfPositivesDS
rttMonLatestJitterOperMaxOfPositivesDS
rttMonLatestJitterOperNumOfPositivesDS
rttMonLatestJitterOperSumOfPositivesDS
rttMonLatestJitterOperSum2PositivesDS
rttMonLatestJitterOperMinOfNegativesDS
rttMonLatestJitterOperMaxOfNegativesDS
rttMonLatestJitterOperNumOfNegativesDS
rttMonLatestJitterOperSumOfNegativesDS
rttMonLatestJitterOperSum2NegativesDS
rttMonLatestJitterOperPacketLossSD
rttMonLatestJitterOperPacketLossDS
rttMonLatestJitterOperPacketOutOfSequence
rttMonLatestJitterOperPacketMIA
rttMonLatestJitterOperPacketLateArrival
rttMonLatestJitterOperOWSumSD
rttMonLatestJitterOperOWSum2SD
rttMonLatestJitterOperOWMinSD
rttMonLatestJitterOperOWMaxSD
rttMonLatestJitterOperOWSumDS
rttMonLatestJitterOperOWSum2DS
rttMonLatestJitterOperOWMinDS
rttMonLatestJitterOperOWMaxDS
rttMonLatestJitterOperNumOfOW
rttMonLatestJitterOperMOS
rttMonLatestJitterOperICPIF

This collector supports purging of collected performance data. This is done on a schedule configured for all of Cisco CNS PerfE, or you can configure a purger just for this collector.

The SaaJitter Collector supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not store the data internally and does not post-process the data after collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

Limitations

None.

SAAUdpEcho Collector

The SaaUdpEcho Collector measures the round trip time for UDP between source and destination devices.

Requirements

A value for **rtr responder** must be configured on the target router.

This collector only works with Cisco devices, which support Service Assurance Agent.

XML Configuration

The following table describes the tags that you can set for the SAAUdpEcho Collector.

Table 6-20 Tags in the XML Configuration for the SAAUdpEcho Collector

XML Tag	Purpose	Mandatory/ Optional
control	Sends a control message to the rtr responder first before the operation. The default is “enabled”.	Optional
index	The index to an existing operation already created on the router.	Optional
packetSize	The payload size of the UDP packet. The default is 64.	Optional
sourcePort	The source UDP port to use. The default is 0 which means that the router will select a random port.	Optional
targetAddress	The IP address of the device on which the SAA operation occurs. The default target address is the one specified outside the <operation> tag	Mandatory
targetPort	The destination UDP port to use. The default target port is the one specified outside the <operation> tag	Mandatory
timeout	The amount of time to wait for a response to the operation.	Optional
tos	The TOS field of the IP packet. The default is 0.	Optional
VrfName	The vrfName in which the saa operation should use.	Optional

Sample XML configuration:

```
<collector name="SAAudpEcho">
<SaaUdpEchoCollector>
<schedule name="S1"/>
<device name="dev1">
<operation name="op1">
<targetAddress>172.29.146.106</targetAddress>
<targetPort>100</targetPort>
<sourcePort>100</sourcePort>
```

```
</operation>
<operation name="op2">
<targetAddress>172.29.146.106</targetAddress>
<targetPort>100</targetPort>
</operation>
</device>
<timeout>PT3S</timeout>
<packetSize>128</packetSize>
<tos>0</tos>
<control>enable</control>
</SaaUdpEchoCollector>
</collector>
```

Data Export

The SaaUdpEcho Collector retrieves measurements about the UdpEcho operation and stores the data in the database. It exports data in Comma Separated Values (CSV) format. Data can be exported at specified frequency or on-demand using the XML interface (<get> <data> tags).

The first line of each data file contains the column headings separated by commas. Each subsequent line contains the following information:

Table 6-21 Order of the Data Exported by the SaaUdpEcho Collector

Order	Parameter
1.	collector name
2.	device name
3.	data collection timestamp
4.	rttMonLatestRttOperCompletionTime
5.	rttMonLatestRttOperSense
6.	rttMonLatestRttOperApplSpecificSense
7.	rttMonLatestRttOperSenseDescription
8.	rttMonLatestRttOperTime

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The SaaUdpEcho Collector supports threshold crossing/clearing alerts (TCAs). TCAs can be configured only for the attribute **rttMonLatestRttOperCompletionTime**. The collector checks for TCAs after each collection period.

The SaaUdpEcho Collector supports purging of collected performance data. This is done on a schedule configured for all of Cisco CNS PerfE or based on an XML configuration for just this collector.

The **SaaUdpEcho Collector** supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not store the data internally and does not post-process the data after collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

Limitations

None.

System Collector

The System Collector monitors the health of the Cisco CNS PerfE host, including disk, CPU and memory usage on the system. The disks that this collector monitors are the hard disks used for the data, the database, and ***<CNS_PerfE_Home>***. In addition, based on the collector configuration, it can also send out fatal system errors and informational messages during system startup and shutdown. It is recommended that every Cisco CNS PerfE host is configured with a System Collector.

Requirements

None.

XML Configuration

No special configuration is required for the System Collector.

```
<das>
      <create>
            <schedule name="S1">
                  <start>2001-08-31T00:00:00.000-08:00</start>
                  <interval>PT1M</interval>
                  <interTaskDelay>PT0.1S</interTaskDelay>
            </schedule>
            <dataHandler name="dh">
                  <ftp>
                        <urlPrefix>ftp://userid:password@ip-address/%2Fvar/tmp/exportSYS</
urlPrefix>
                  </ftp>
            </dataHandler>
            <threshold name="th1">
                  <attribute name="sys.disk.db.usedPercentage">
                        <raiseOperation>GT</raiseOperation>
                        <raiseValue>10</raiseValue>
                        <clearOperation>LT</clearOperation>
                        <clearValue>30</clearValue>
                        <level>minor</level>
                  </attribute>
            </threshold>
            <notifier name="trapNot">
                  <trap>
                        <ipaddress>10.77.11.125</ipaddress>
                  </trap>
            </notifier>
            <notifier name="SYScnsNot">
                  <cns>
                        <subject>SYS-cns</subject>
                  </cns>
            </notifier>

            <collector name="sys">
                  <SystemCollector>
                        <schedule name="S1"/>
                        <threshold name="th1"/>
                        <notifier name="SYScnsNot"/>
                        <notifier name="trapNot"/>
                        <dataHandler name="dh"/>
                  </SystemCollector>
```

```
            </collector>
      </create>
      <start>
            <collector name="sys"/>
      </start>
</das>
```

Data Export, Threshold Crossing Alerts, and Purging Data

The System Collector exports data in Comma Separated Values (CSV) format. Data can be exported at specified frequency or on-demand using the XML interface (<get> <data> tags). The first line of each exported file contains the column headings separated by commas.

Order of the data exported by the System Collector

1. collector name
2. collectionTime,
3. memoryAvailableMega
4. memoryUsedPercentage
5. diskHomeAvailableMega
6. diskHomeUsedPercentage
7. diskDBAvailableMega
8. diskDBUsedPercentage
9. diskDataAvailableMega
10. diskDataUsedPercentage
11. cpuLoadPercentage
12. javaMemoryAvailableMega
13. javaMemoryUsedPercentage

The System Collector supports threshold crossing/clearing alerts (TCAs). TCAs are configured for specific attributes. The collector checks for TCAs after each collection period. Attribute names on which thresholds can be configured are given below.

The System Collector supports purging of collected performance data. This can be done based on purge parameters in das.properties or you can configure a purger for the System Collector.

The System Collector supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not post-process the data after collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

The System Collector supports thresholding on the following attributes:

Table 6-22 System Collector Threshold Attributes

Attribute	Description
sys.memory.availableMega	Memory available on the machine (in megabytes).
sys.memory.usedPercentage	Memory used on the machine (percentage).
sys.disk.home.availableMega	Disk space available for the home directory (in megabytes).
sys.disk.home.usedPercentage	Disk space used for the home directory (percentage).

Table 6-22 System Collector Threshold Attributes (continued)

Attribute	Description
sys.disk.db.availableMega	Disk space available for the database directory (in megabytes).
sys.disk.db.usedPercentage	Disk space used for the database directory (percentage).
sys.disk.data.availableMega	Disk space available for the data directory (in megabytes).
sys.disk.data.usedPercentage	Disk space used for the data directory (percentage).
sys.cpu.loadPercentage	CPU load on the machine (percentage).
java.memory.availableMega	Memory available on java virtual machine (in megabytes)
java.memory.usedPercentage	Memory used on java virtual machine (percentage).

Limitations

None.

VoiceCallStatistics Collector

The VoiceCallStatistics Collector (VCS Collector) collects and processes data from Voice Call Performance Statistics on Cisco Gateways and Cisco VoIP Internal Error Codes projects.

The Voice Call Performance Statistics provides summarized call statistics and call accounting statistics. It provides call statistics at the following levels:

voice port, trunk group, ip, pstn, gw

and exports the statistics to the Cisco CNS PerfE host via FTP.

The Cisco VoIP Internal Error Codes feature in IOS adds error counters for each subsystem. These counters will be generated at the same interval as the TH counter generation and both data sets will be transmitted in a single file to Cisco CNS PerfE host at each interval.

The VoiceCallStatistics Collector is configured for receiving these statistics. It stores all the statistics and forwards the data to the northbound application through a consistent interface.

Requirements

Cisco CNS PerfE location needs to be configured in a CLI in the Voice Call Performance Statistics device so that its data files would be transferred to the specified directory in the Cisco CNS PerfE server.

The Cisco CNS PerfE location directory needs to have write permission so that NE can push the file to the same location and Cisco CNS PerfE can remove these files after they are processed.

The router should be configured with the same name as that used in <create> <device> tags in Cisco CNS PerfE.

XML Configuration

The following XML example shows the typical configuration for the VoiceCallStatistics Collector.

```
<schedule name="VCSSchedule">
```

```
      <interval>PT15M</interval>
</schedule>

<!-this device name should be the same as the router name configured on the device-->
<device name="gw1">
      …
</device>

<collector name="VCSCollector">
      <VoiceCallStatisticsCollector>
            <schedule name="VCSSchedule"/>
            <device name="gw1"/>
            <localDir>/export/VCSdir</localDir>
      </VoiceCallStatisticsCollector>
</collector>
```

Note localDir specifies the location on the Cisco CNS PerfE host where the NE FTPs files to Cisco CNS PerfE. This is the directory that the routers are configured with. Routers export files to this location on Cisco CNS PerfE host. Cisco CNS PerfE reads files from this directory and processes them.

Data Export

The VoiceCallStatisitcs Collector exports data in XML format. Data can be exported at a specified frequency or on-demand using the XML interface (<get> <data> tags).

Below is a typical example of exported data by the VoiceCallStatistics Collector:

```
<VCSStats>
      <collector name="VCSC">
            <device name="as5400">
                  <collectionTime time="2003-02-27T05:28:30.000Z">
                        <csrStats startTime="2003-02-26T23:25:58.000Z"
endTime="2003-02-26T23:25:59.000Z">
                              <gw>

                                    <inCalls>100</inCalls>
                                    <inAns>100</inAns>
                                    <inRej>0</inRej>
                                    <outCalls>100</outCalls>
                                    <outAns>100</outAns>
                                    <outFail>0</outFail>
                                    <inSeizureDur>null</inSeizureDur>
                                    <outSeizureDur>507</outSeizureDur>
                                    <inConnDur>504</inConnDur>
                                    <outConnDur>504</outConnDur>
                                    <origDisconn>0</origDisconn>
                                    <inAnsAbnormal>0</inAnsAbnormal>
                                    <outAnsAbnormal>0</outAnsAbnormal>
                                    <inMcd>0</inMcd>
                                    <outMcd>0</outMcd>
                                    <inPDD>4630</inPDD>
                                    <outPDD>2810</outPDD>
                                    <inSetupDelay>1730</inSetupDelay>
                                    <outSetupDelay>1730</outSetupDelay>
                                    <inDiscCauseCode>
                                          <count code="16">100</count>
                                    </inDiscCauseCode>
                                    <outDiscCauseCode>
                                          <count code="16">100</count>
```

```
                                    </outDiscCauseCode>


                                </gw>
                                <ip>
                                ....
                                </ip>
                                <pstn>
                                ....
                                </pstn>
                                <tg id="orig2">
                                ....
                                </tg>
                                <vp id="6/0:D" tgid="t1">
                                ....
                                </vp>
                          </csrStats>
                          <casrStats startTime="2003-02-26T06:49:44.000Z"
endTime="2003-02-26T06:49:45.000Z">
                                <ml id="h323">

                                      <acctPassCriteria>1</acctPassCriteria>
                                      <pstnInPass>0</pstnInPass>
                                      <pstnInFail>0</pstnInFail>
                                      <pstnOutPass>0</pstnOutPass>
                                      <pstnOutFail>0</pstnOutFail>
                                      <ipInPass>0</ipInPass>
                                      <ipInFail>0</ipInFail>
                                      <ipOutPass>0</ipOutPass>
                                      <ipOutFail>0</ipOutFail>
                                </ml>
                                <ml id="ddtest">
                                      ....
                                      ....
                                </ml>
                                ....
                                ....

                          </casrStats>
                          <iecStats startTime="2003-02-26T06:49:44.000Z"
endTime="2003-02-26T06:49:45.000Z">
                                <count subsysid ="3" code="5">1</count>
            .....
                          </iecStats>
                    </collectionTime>
            </device>
      </collector>
</VCSStats>
```

Similarly when multiple files are collected in one interval the export format will be as follows:

Suppose the two files are collected in an interval as FN1, FN2 at the same time T1 (from the same device).

FN1 has section CSR1, CASR1, IEC1.

FN2 has sections CSR2, CASE2, IEC2.

The output from CNS-PerfE are as follows:

```
<VCSStats>
      <collector .....>
            <device ...>
                  <collectionTime T1>
                        <csrStats for CSR1>       ...        </csrStats>
```

```
                        <csrStats for CSR2>       ....       </csrStats>
                        <casrStats for CASR1>   ....       </casrStats>
                        <casrStats for CASR2>   ....       </casrStats>
                        <iecStats for IEC1>         ...         </iecStats>
                        <iecStats for IEC2>         ...         </iecStats>
                    </collectionTime>
            ...
            ...
</VCSStats>
```

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The VoiceCallStatistics Collector supports threshold crossing/clearing alerts (TCAs). TCAs are configured for specific attributes. The collector checks for TCAs after each collection period. Attribute names on which thresholds can be configured are given below.

The VoiceCallStatistics Collector supports purging of collected performance data. This can be done based on purge parameters in das.properties or you can configure a purger for the VoiceCallStatistics Collector.

The VoiceCallStatistics Collector supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not store the data internally and does not post-process the data after data collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.

The following are the supported threshold attribute names:

GW attribute names gw.<tagname>	IP attribute names ip.<tagname>
gw.inCalls	ip.inCalls
gw.inAns	ip.inAns
gw.inRej	ip.inRej
gw.outCalls	ip.outCalls
gw.outAns	ip.outAns
gw.outFail	ip.outFail
gw.inSeizureDur	ip.inSeizureDur
gw.outSeizureDur	ip.outSeizureDur
gw.inConnDur	ip.inConnDur
gw.outConnDur	ip.outConnDur
gw.origDisconn	ip.origDisconn
gw.inAnsAbnormal	ip.inAnsAbnormal
gw.outAnsAbnormal	ip.outAnsAbnormal
gw.inMcd	ip.inMcd
gw.outMcd	ip.outMcd
gw.inPDD	ip.inPDD
gw.outPDD	ip.outPDD

GW attribute names gw.<tagname>	IP attribute names ip.<tagname>
gw.inSetupDelay	ip.inSetupDelay
gw.inSetupDelay	ip.inSetupDelay
gw.outSetupDelay	ip.outSetupDelay
	ip.lostPkt
	ip.latency
	ip.jitter
gw.inDiscCauseCode.<causecode#>	ip.inDiscCauseCode.<causecode#>
gw.outDiscCauseCode.<causecode#>	ip.outDiscCauseCode.<causecode#>

Similarly, for pstn, the names will be as follows:

```
pstn.<tagname>
```

For trunkgroup data, the names will be as follows:

```
tg.<tgid>.<tagname>
```

For Voice port data, the names will be as follows:

```
vp.<vpid>.<tgid>.<tagname>.[code]
```

For IEC data, the names will be as follows:

```
iec.<subsystemid>.<errcode>
```

Below, sample threshold attribute names and their intended behavior are explained:

- tg.10.inCalls: match inCalls values for trunk group whose id is "10".
- tg..inCalls: match inCalls values for all trunk groups.
- vp.1.10.inCalls: match inCalls for the voice port whose id is "1" and trunkgroupid is "10".
- vp.1..inCalls: match inCalls for the voice port whose id is "1".
- vp…inCalls: match inCalls for all voice ports.

The following names will be supported for threshold names for call accounting statistics:

```
<methodList>.<methodList_name>.<threshold attributes> [ methodList is the name of the list
(e.g. radius123) ]
```

Example:

ml.radius123.acctPassCriteria

ml.radius123.pstnInPass

ml.radius123.pstnInFail

ml.radius123.pstnOutPass

ml.radius123.pstnOutFail

ml.radius123.ipInPass

ml.radius123.ipInFail

ml.radius123.ipOutPass

ml.radius123.ipOutFail

VoipDAS Collector

The VoipDAS Collector collects correlated call data from multiple Cisco CNS PerfE hosts, correlates four call legs for each call. Because of the distributed nature of Cisco CNS PerfE, one Cisco CNS PerfE may not have the call detail record for all four of the call legs. For example, call legs 1 and 2 are collected by Cisco CNS PerfE A, while call legs 3 and 4 are collected by Cisco CNS PerfE B. In cases like this, each Cisco CNS PerfE correlates the data it has. Cisco CNS PerfE A correlates call leg 1 and call leg 2 while Cisco CNS PerfE B correlates call leg 3 and 4. In this case, VoipDas Collector is similar to any other collector in Cisco CNS PerfE and treats the underlying Cisco CNS PerfE systems as the data sources for call correlated data.

Multiple Cisco CNS PerfE hosts are specified in the configuration as **urlPrefixes**, that is, each URL Prefix identifies the username and password to access that particular Cisco CNS PerfE host, and the data directory in which correlated files are stored. The VoipDAS Collector picks up files from this directory for correlation and deletes the files from the data source after the files are retrieved.

Requirements

If an external network device will export data to Cisco CNS PerfE host, the FTP server should be configured and running on the same host where Cisco CNS PerfE is running. Refer to the Admin User Guide manual on how to configure a FTP server for the platform you are using.

Note that in CNS PerfE, the **dasadmin** account is used by the VoIPDAS Collector to ftp data to the CNS PerfE host. If CNS PerfE is installed under **/opt**, ftp fails, because **dasadmin** cannot access the **/opt** directory. As a result, the **/etc/ftpaccess** file should have the **dasadmin** user in the access list as follows.

```
realuser root dasadmin
```

This enables **dasadmin** user to access root directories, viz. /opt.

XML Configuration

The following table describes the tags that you can set for the VoipDAS Collector.

Table 6-23 Tags in the XML Configuration for the VoipDAS Collector

XML Tag	Purpose	Mandatory/ Optional
urlPrefixes	Identifies the username and password to access that particular Cisco CNS PerfE host, and the data directory in which correlated files are stored.	Mandatory

Sample XML configuration with FTP:

```
<collector name="v1">
      <VoipDasCollector>
            <schedule name="S1"/>
            <urlPrefix>ftp://user:passwd@DASA/%2Fopt/CSCOdas/data/VoipCorrelator/</urlPref
ix>
```

```
            <urlPrefix>ftp://user:passwd@DASB/%2Fopt/CSCOdas/data/VoipCorrelator/</urlPref
ix>
      </VoipDasCollector>
</collector>
```

Sample XML configuration with SFTP:

```
<collector name="v1">
      <VoipDasCollector>
            <schedule name="S1"/>
            <urlPrefix>ftp://user:passwd@DASA/%2Fopt/CSCOdas/data/VoipCorrelator/</urlPref
ix>
            <urlPrefix>sftp://user:passwd@DASB/%2Fopt/CSCOdas/data/VoipCorrelator/</urlPre
fix>
      </VoipDasCollector>
</collector>
```

Data Export, Threshold Crossing Alerts, Purging Data, and Hotspot Polling

Refer to the RADIUS Collector sections: "Data Export" section on page 6-74 and "Threshold Crossing Alerts, Purging Data, and Hotspot Polling" section on page 6-78. The data files from VoipDAS Collector will be generated in the ***<CNS_PerfE_Home>*/data/VoipDasCollector** directory.

Limitations

Because the VoipDAS Collector does very large FTP operations and correlates CDR files, it is resource-intensive. It is recommended that you install the VoipDAS Collector only on a 2nd-tier Cisco CNS PerfE system (see Figure 1-1 on page 1-2 for more information on 2nd-tier systems).

VPDNCustomer Collector

VPDN (Virtual Private Dialup Network) is a feature of Cisco IOS. VPDN handles the forwarding of PPP links from an Internet Provider (ISP) to a Home Gateway.

The VPDN tunnel's user information is a manageable entity in VPDN.

VPDNCustomer Collector is a collector in Cisco CNS PerfE, which collects data about the tunnel information for a customer using SNMP and reports the following information per customer:

- totalTunnels –Total number of tunnels assigned
- totalActiveSessions –Total number of active sessions currently in all the tunnels
- totalDeniedUsers –Total number of denied users (cumulative)

XML Configuration

```
<das>
      <load>
            <mib name="CISCO-VPDN-MGMT-MIB.my"/>
      </load>
      <create>
            <schedule name="VPDNSchedule">
                  <start>2002-03-06T00:00:00.000-08:00</start>
                  <interval>PT2M</interval>
```

```
            </schedule>

            <device name="VPDNDevice">
                  <snmp>
                        <ipaddress>10.0.9.25</ipaddress>
                        <readCommunity>public</readCommunity>
                        <writeCommunity>private</writeCommunity>
                  </snmp>
            </device>

            <dataHandler name="VPDNExport">
                  <url>
                        <prefix>ftp://user:password@b-netrax1/%2Fdata/VPDNPeriodic</prefix
>
                  </url>
            </dataHandler>

            <threshold name="VPDNThreshold">
                  <attribute name="totalActiveSessions.sbc">
                        <raiseOperation>GT</raiseOperation>
                        <raiseValue>30</raiseValue>
                        <clearOperation>LT</clearOperation>
                        <clearValue>100</clearValue>
                        <level>critical</level>
                  </attribute>
            </threshold>

            <notifier name="VPDNNotifyCNS">
                  <cns>
                        <subject>cns.cisco.das.listener</subject>
                  </cns>
            </notifier>

            <collector name="VPDNCollect">
                  <VPDNCustomerCollector>
                        <schedule name="VPDNSchedule"/>
                        <device name="VPDNDevice"/>
                        <threshold name="VPDNThreshold"/>
                        <notifier name="VPDNNotifyCNS"/>
                        <dataHandler name="VPDNExport"/>

                  </VPDNCustomerCollector>
            </collector>
      </create>
      <start>
            <collector name="VPDNCollect"/>
      </start>
</das>
```

Data Export

Data is exported in CSV format as follows:

- The first line of the file contains the version number.
- The second line of the file contains the header.
- The rest of the file contains the data.

Sample data:

```
Version = 1.0
```

```
CollectorName, DeviceName, CustomerName, Timestamp, TotalTunnels, TotalActiveSessions,
TotalDeniedUsers
VPDNCollect,10.0.9.25,aol,2002-11-22T08:38:00.017Z,2,40,43
VPDNCollect,10.0.9.25,att,2002-11-22T08:38:00.017Z,3,18,33
VPDNCollect,10.0.9.25,sbc,2002-11-22T08:38:00.017Z,4,41,35
```

Threshold Crossing Alerts, Purging Data, and Hotspot Polling

The VPDNCustomer Collector supports threshold crossing/clearing alerts (TCAs). TCAs are configured for specific attributes. The collector checks for TCAs after each collection period.

Thresholds can be applied to *totalActiveSessions*, *totalDeniedUsers*. *CustomerName* can be appended to the attribute name to support threshold for a specific customer.

The threshold attribute names can specified as follows:

totalActiveSessions.<customername>

or

totalActiveSessions

totalDeniedUsers.<customername>

or

totalDeniedUsers

For example, if the threshold attribute name is *totalActiveSessions.aol*, then the threshold is applied to the total number of active sessions for aol customer. If threshold attribute name is *totalActiveSessions*, then the threshold is applied to all customers. This collector supports purging of collected performance data. This can be done based on purge parameters in **das.properties** or you can configure a purger for the VPDNCustomer Collector.

It supports hotspot polling. If the collector is configured for hotspot polling, then Cisco CNS PerfE does not store the data internally and does not post-process the data after data collection. As soon as the data is retrieved, the data is written to the CNS Integration Bus based on the configuration.