# Cisco Prime Performance Manager 1.6 Integration Developer Guide

August 28, 2015

**Cisco Systems, Inc.**
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.

*Cisco Prime Performance Manager 1.6 Integration Developer Guide*

# CONTENTS


# Getting Started with Cisco Performance Management Reports

This guide describes the organization and structure of Prime Performance Manager reports so that you can extend the built-in reports that come with Prime Performance Manager and develop your own customized service reports.

This chapter contains:



## XML-Based Service Reports

Prime Performance Manager reports are coded in XML. Out-of-the-box, Prime Performance Manager includes more than 4400 reports that you can use as examples. You can copy an example report to your working directory, modify the XML, and then test and debug your report

The built-in reports display data reports on:

- Application traffic
- Applications
- Availability of device interfaces, MPLS tunnels, pseudowires, and SNMP devices
- Compute technologies including ESXi, Hyper-V, KVM, UCS clusters, and others.
- Data Network Services
- IP Quality of Service (QoS)
- IP Protocol performance
- IP Service-Level Agreement (SLA) statistics
- Layer 2 protocols
- Mobile IOS Statistics
- Mobile StarOS Statistics

- Mobile StarOS CDMA KPI
- Mobile StarOS CDMA statistics
- Mobile StarOS KPI
- Mobile StarOS statistics
- NetFlow
- NetFlow CGNAT
- Network and network services
- Prime Performance Manager system reports
- Resource Utilization, such as CPU and memory utilization
- Security
- Storage (Netapp, EMC, and Fibre Channel)
- Transport statistics, such as ATM statistics, Ethernet Virtual Circuit (EVC) statistics, and so on
- Video Broadcast

For details, see the *Cisco Prime Performance Manager 1.7 Data Sheet* on www.cisco.com. You should:

1. Go to http://www.cisco.com/go/performance.
2. Choose **Product Literature > Data Sheets**.

# Key Components of the Report Writing Interface

The XML files, MIBs, and configuration files that you need to develop reports are located on the Prime Performance Manager Gateway server.

This guide tells you how to use:

- **Supported MIBs**—Prime Performance Manager provides support for over 600 Cisco and industry standard MIBs that you can use to develop reports
- **Capability Files**—Capability files let you define which MIBs are used in reports and which MIB variables are polled. There are two capability files:
  - SystemCapability.xml—Precoded system capability file. This file specifies the capabilities for the reports provided with the Prime Performance Manager system. Do not change this file.
  - UserCapability.xml—System capability file for user-developed reports. If you need to add or revise the reporting capabilities, specify your changes in this file.
- **Prepackaged XML Reports**—The Prime Performance Manager application provides over 380 XML report files that you can use as a model for your own reports.
- **Properties Files**—Each XML report file has a corresponding properties file that maps to the report XML file and defines variables used in the online reports and CSV reports.
- **Report Macros**—A collection of SNMP macros that you can call from your XML report code and UserCapability.xml file to perform processing tasks.

  For example, an IpAddress() macro is provided to convert a specified object to an IP address.
- **BQL Files**—BQL Files enable cross-launching of your reports on Cisco Prime Network clients.
- **Online Help**—Includes system-generated help for your reports and a Reports Help page.

# Directory Locations of Report Files and Related Files

Table 1-1 lists the locations of the reports and related files.

*Table 1-1 Directory Locations for Prime Performance Manager Reports and Support Files*

| Files | Location |
|---|---|
| Performance reports (5-minute, 15-minute, hourly, daily, weekly, monthly) exported to CSV. | /opt/CSCOppm-gw/reports |
| Gateway log | opt/CSCOppm-gw/logs |
| Unit log | opt/CSCOppm-unit/logs |
| MIBs | /opt/CSCOppm-gw/etc/mibs |
| Capability Files | /opt/CSCOppm-gw/etc/ SystemCapability.xml<br>/opt/CSCOppm-gw/etc/ UserCapability.xml |
| System XML reports/properties files | /opt/CSCOppm-gw/etc/pollers/system |
| User XML reports/properties files | /opt/CSCOppm-gw/etc/pollers/user |
| BQL Files | /opt/CSCOppm-gw/etc/bql/xl |
| Report Macros | Precompiled on the Prime Performance Manager gateway. See Chapter B, "Reports Macro Reference" for reference information on the macros. |

# Report Processing Overview

The Prime Performance Manager processes reports as follows:

1. Prime Performance Manager polls the devices in the network inventory based on:
   - The MIBs selected for polling in Prime Performance Manager reports.
   - Filtering specified in the SystemCapability.xml file and in the UserCapability.xml file.
   - The filtering process queries polled devices as to whether they actually support the MIB being used, and if the MIB is supported. It restricts polling to MIB objects that actually have table data and which meet other specified criteria. This ensures that Prime Performance Manager does not perform unnecessary polling.
2. Based on what MIB variables are polled by the system XML reports and by user-defined reports, Prime Performance Manager creates a virtual database table that contains the polled data.

   This results in faster processing of polling data and allows reports to be displayed quickly.
3. Based on what you specify in your report XML, the system processes the data returned by the polling. You can use predefined reporting macros to manipulate the data.

   For example, you can convert a value to a percentage.
4. Based on macro calls in the system reports and user-defined reports, Prime Performance Manager modifies the virtual tables. For example, two tables can be joined or data can be selected for inclusion in table rows.
5. Reports that contain data can be viewed when users select them from the Prime Performance Manager Reports tree. The appearance of the reports is customizable in the report XML.

6. At the end of each reporting period configured for the server, the system saves the virtual table data to the Prime Performance Manager database.

The ability to customize data polling and report display provides you with a flexible and powerful way to report data to users. In the tutorial chapter of this guide,Chapter 2, "Writing a Report," we'll walk you through the coding of a typical report, the cpu.xml report, and show you how you can modify a sample report to develop your own reports.

# Online Reports Help

Prime Performance Manager provides an extensive help system to help you develop reports, including autogenerated help for your report, and a Reports Help page.

This section contains:



## Auto-Generated Help for Your Report

When you write and enable a report, Prime Performance Manager automatically creates online help for your project. It also allows you to write and publish a customized help file for it. Prime Performance Manager rebuilds the report help files once every night. You can also manually regenerate the report help by issuing the **ppm docreps** command from the gateway CLI.

The autogenerated help includes:

- **Links to the Report Definition File**— Click on the XML filename to view the XML definition. You can view the definition in the PPM Viewer (as straight ASCII text). You can also use the browser's frame source viewer, which provides a color-coded view that highlights XML keywords and coding elements.
- **Custom Help**—Click on the Custom Help link to display customized help for the report.
- **Links to the MIBs Used in the Report**. Click a MIB filename to display the MIB.

Example 1-1 shows the online help for the cpu.xml report.

***Example 1-1 System-Generated Help for the cpu.xml Report***

```
============================================================
Definition File:
   cpu.xml   (PPM     Viewer)
   cpu.xml   (Browser Viewer - Use View Page Source Menu For Color Coded View)
============================================================

Online Help
Custom Help

MIBs Used:
   CISCO-PROCESS-MIB.my

MIBs Used:
   ENTITY-MIB.my

MIBs Used:
   HOST-RESOURCES-MIB.my
```

```
MIBs Used:
   UCD-SNMP-MIB.my

-------------------------------------------------------------

Variables Collected:

      cpmCPUTotalTable = poll("cpmCPUTotalIndex,
                               cpmCPUTotalPhysicalIndex,
                               cpmCPUTotal5minRev,
                               cpmCPUTotal1minRev");

Variables Collected:

      cpmCPUThresholdTable = poll("cpmCPUTotalIndex,
                                   cpmCPUThresholdClass,
                                   cpmCPURisingThresholdValue,
                                   cpmCPUFallingThresholdValue");

-------------------------------------------------------------

Variables Collected:

      cpmCPUTotalTable = poll("cpmCPUTotalIndex,
                               cpmCPUTotalPhysicalIndex,
                               cpmCPUTotal5min,
                               cpmCPUTotal1min");

Variables Collected:

      cpmCPUThresholdTable = poll("cpmCPUTotalIndex,
                                   cpmCPUThresholdClass,
                                   cpmCPURisingThresholdValue,
                                   cpmCPUFallingThresholdValue");

-------------------------------------------------------------

Variables Collected:

       hrProcessorTable = poll("hrDeviceIndex,
                                hrProcessorLoad");

Variables Collected:

       hrDeviceTable = poll("hrDeviceIndex,
                             hrDeviceDescr");

-------------------------------------------------------------

Variables Collected:

       ssCpuStats = poll("ssCpuRawUser,
                          ssCpuRawNice,
                          ssCpuRawSystem,
                          ssCpuRawIdle,
                          ssCpuRawWait,
                          ssCpuRawKernel,
                          ssCpuRawInterrupt,
                          ssCpuRawSoftIRQ,
                          ssRawInterrupts,
                          ssRawContexts");

=============================================================================
```

```
CSV File Format For cpu.xml Reports
Report ID: CPUOLD                              MIB Used: CISCO_PROCESS_MIB_CPUOLD
================================================================================

1 CSV Filename Prefix: CPU
--------------------------------------------------------------------------------
 1 TimeStamp                              TimeStamp
 2 Node                                   fqdnid
 3 Slot                                   CPUSlot
 4 Number                                 CPUNum
 5 Description                            CPUDescr
 6 CPUUtilMax5min                         Max(cpmCPUTotal5min / 100)
 7 CPUUtilAvg5min                         Avg(cpmCPUTotal5min / 100)
 8 CPUUtilMax1min                         Max(cpmCPUTotal1min / 100)
 9 CPUUtilAvg1min                         Avg(cpmCPUTotal1min / 100)
10 CPURisingThreshold                     cpmCPURisingThresholdValue / 100
11 CPUFallingThreshold                    cpmCPUFallingThresholdValue / 100
12 ProcessorIndex                         cpmCPUTotalIndex

================================================================================
Web Reports For cpu.xml Reports
Report ID: CPUOLD                              MIB Used: CISCO_PROCESS_MIB_CPUOLD
================================================================================

1 CPU Activity Type Percentage
--------------------------------------------------------------------------------
User Code                                UserCPUPercent
System Code                              SystemCPUPercent
Kernel Code                              KernelCPUPercent
Nice Code                                NiceCPUPercent
Idle                                     IdleCPUPercent
IO Wait                                  WaitCPUPercent
H/W Interrupts                           HWInterruptCPUPercent
S/W Interrupts                           SWInterruptCPUPercent
Device                                   fqdnid
Interrupts                               Interupts
Context Switches                         ContextSwitches

2 CPU Utilization
--------------------------------------------------------------------------------
Average Utilization                      Avg(cpmCPUTotal5min / 100)
Peak Utilization                         Max(cpmCPUTotal5min / 100)
Device                                   fqdnid
Slot                                     CPUSlot
CPU                                      CPUNum
CPU Description                          CPUDescr
Avg                                      Avg(cpmCPUTotal1min / 100)
Peak                                     Max(cpmCPUTotal1min / 100)


================================================================================
CSV File Format For cpu.xml Reports
Report ID: UCD_CPU                             MIB Used: UCD_CPU
================================================================================

1 CSV Filename Prefix: UCD_CPU_ACTIVITY
--------------------------------------------------------------------------------
 1 TimeStamp                              TimeStamp
 2 Node                                   fqdnid
 3 UserPercent                            Avg(userCPUTicks        / totalTicks)
 4 SystemPercent                          Avg(systemCPUTicks      / totalTicks)
 5 KernelPercent                          Avg(kernelCPUTicks      / totalTicks)
 6 NicePercent                            Avg(niceCPUTicks        / totalTicks)
 7 IdlePercent                            Avg(idleCPUTicks        / totalTicks)
```

```
 8 IOWaitPercent                          Avg(waitCPUTicks        / totalTicks)
 9 H/WInterruptsPercent                   Avg(hwInterruptCPUTicks / totalTicks)
10 S/WInterruptsPercent                   Avg(swInterruptCPUTicks / totalTicks)
11 InterruptsCount                        Sum(ssRawInterrupts.delta())
12 ContextSwitchesCount                   Sum(ssRawContexts.delta())

================================================================================
Web Reports For cpu.xml Reports
Report ID: UCD_CPU                           MIB Used: UCD_CPU
================================================================================

1 CPU Activity Type Percentage
--------------------------------------------------------------------------------
User Code                                 Avg(userCPUTicks        / totalTicks)
System Code                               Avg(systemCPUTicks      / totalTicks)
Kernel Code                               Avg(kernelCPUTicks      / totalTicks)
Nice Code                                 Avg(niceCPUTicks        / totalTicks)
Idle                                      Avg(idleCPUTicks        / totalTicks)
IO Wait                                   Avg(waitCPUTicks        / totalTicks)
H/W Interrupts                            Avg(hwInterruptCPUTicks / totalTicks)
S/W Interrupts                            Avg(swInterruptCPUTicks / totalTicks)
Device                                    fqdnid
Interrupts                                Sum(ssRawInterrupts.delta())
Context Switches                          Sum(ssRawContexts.delta())

2 CPU Utilization
--------------------------------------------------------------------------------
Average Utilization                       Avg(cpmCPUTotal5min / 100)
Peak Utilization                          Max(cpmCPUTotal5min / 100)
Device                                    fqdnid
Slot                                      CPUSlot
CPU                                       CPUNum
CPU Description                           CPUDescr
Avg                                       Avg(cpmCPUTotal1min / 100)
Peak                                      Max(cpmCPUTotal1min / 100)


================================================================================
CSV File Format For cpu.xml Reports
Report ID: CPU                               MIB Used: CISCO_PROCESS_MIB_CPUREV
================================================================================

1 CSV Filename Prefix: CPU
--------------------------------------------------------------------------------
 1 TimeStamp                              TimeStamp
 2 Node                                   fqdnid
 3 Slot                                   CPUSlot
 4 Number                                 CPUNum
 5 Description                            if((cpuDescr == ""), "NoDescription", cpuDescr)
 6 CPUUtilMax5min                         Max(cpmCPUTotal5minRev / 100)
 7 CPUUtilAvg5min                         Avg(cpmCPUTotal5minRev / 100)
 8 CPUUtilMax1min                         Max(cpmCPUTotal1minRev / 100)
 9 CPUUtilAvg1min                         Avg(cpmCPUTotal1minRev / 100)
10 CPURisingThreshold                     cpmCPURisingThresholdValue / 100
11 CPUFallingThreshold                    cpmCPUFallingThresholdValue / 100
12 ProcessorIndex                         cpmCPUTotalIndex

================================================================================
Web Reports For cpu.xml Reports
Report ID: CPU                               MIB Used: CISCO_PROCESS_MIB_CPUREV
================================================================================

1 CPU Activity Type Percentage
--------------------------------------------------------------------------------
```

```
User Code                                Avg(userCPUTicks        / totalTicks)
System Code                              Avg(systemCPUTicks      / totalTicks)
Kernel Code                              Avg(kernelCPUTicks      / totalTicks)
Nice Code                                Avg(niceCPUTicks        / totalTicks)
Idle                                     Avg(idleCPUTicks        / totalTicks)
IO Wait                                  Avg(waitCPUTicks        / totalTicks)
H/W Interrupts                           Avg(hwInterruptCPUTicks / totalTicks)
S/W Interrupts                           Avg(swInterruptCPUTicks / totalTicks)
Device                                   fqdnid
Interrupts                               Sum(ssRawInterrupts.delta())
Context Switches                         Sum(ssRawContexts.delta())

2 CPU Utilization
-------------------------------------------------------------------------------
Average Utilization                      Avg(cpmCPUTotal5minRev / 100)
Peak Utilization                         Max(cpmCPUTotal5minRev / 100)
Device                                   fqdnid
Slot                                     CPUSlot
CPU                                      CPUNum
CPU Description                          if((cpuDescr == ""), "NoDescription", cpuDescr)
Avg                                      Avg(cpmCPUTotal1minRev / 100)
Peak                                     Max(cpmCPUTotal1minRev / 100)


===============================================================================
CSV File Format For cpu.xml Reports
Report ID: HOST_RESOURCES_CPU                 MIB Used: HOST_RESOURCES_CPU_MIB
===============================================================================

1 CSV Filename Prefix: CPU
-------------------------------------------------------------------------------
 1 TimeStamp                               TimeStamp
 2 Node                                    fqdnid
 3 Slot                                    0
 4 Number                                  hrDeviceIndex
 5 Description                             hrDeviceDescr
 6 CPUUtilMax5min                          Max(oneMinUtil)
 7 CPUUtilAvg5min                          Avg(oneMinUtil)
 8 CPUUtilMax1min                          Max(hrProcessorLoad/100)
 9 CPUUtilAvg1min                          Avg(hrProcessorLoad/100)
10 CPURisingThreshold                      0
11 CPUFallingThreshold                     999
12 ProcessorIndex                          hrDeviceIndex

===============================================================================
Web Reports For cpu.xml Reports
Report ID: HOST_RESOURCES_CPU                 MIB Used: HOST_RESOURCES_CPU_MIB
===============================================================================

1 CPU Activity Type Percentage
-------------------------------------------------------------------------------
User Code                                Avg(userCPUTicks        / totalTicks)
System Code                              Avg(systemCPUTicks      / totalTicks)
Kernel Code                              Avg(kernelCPUTicks      / totalTicks)
Nice Code                                Avg(niceCPUTicks        / totalTicks)
Idle                                     Avg(idleCPUTicks        / totalTicks)
IO Wait                                  Avg(waitCPUTicks        / totalTicks)
H/W Interrupts                           Avg(hwInterruptCPUTicks / totalTicks)
S/W Interrupts                           Avg(swInterruptCPUTicks / totalTicks)
Device                                   fqdnid
Interrupts                               Sum(ssRawInterrupts.delta())
Context Switches                         Sum(ssRawContexts.delta())

2 CPU Utilization
```

```
--------------------------------------------------------------------------------
Average Utilization                     Avg(oneMinUtil)
Peak Utilization                        Max(oneMinUtil)
Device                                  fqdnid
Slot                                    0
CPU                                     hrDeviceIndex
CPU Description                         hrDeviceDescr
Avg                                     Avg(hrProcessorLoad/100)
Peak                                    Max(hrProcessorLoad/100)


Implementation Notes for cpu
================================================

Steps for setting CPU to be monitored in snmpd.conf file:

net-snmp should be installed before doing the following steps.

Linux
-----

   1. Open the file /etc/snmp/snmpd.conf

   2. Add the following line

         load  90 80 70

   3. Save the file and restart as follows

         /etc/init.d/snmpd restart

Solaris
-------

   1. Open the file  /etc/sma/snmp/snmpd.conf

   2. Add the following line

         load  90 80 70

   3. Save the file and restart as follows

         /etc/init.d/init.sma restart
```

# Reports Help

To display the online reports help, select the following:

- **Help > READMEs and CLI Commands**—Displays README, CHANGEs, Device Info, CLI Commands, CLI Commands Help, Release Notes, Quick Start.
- **Help > Reports**—Displays System Reports Readme, User Reports Readme, Reports XML Definition, Reports List Readme, IETF RFCs, SNMP MIBS, System Capability Definitions, User Capability Definitions.

# What You Can Specify

Aside from the MIBs that are polled and MIB variables reported, you can specify report views, report time intervals, sort order for data.

## Report Views: Graphs, Tables, and CSV Files

The XML interface lets you code reports to provide users with:

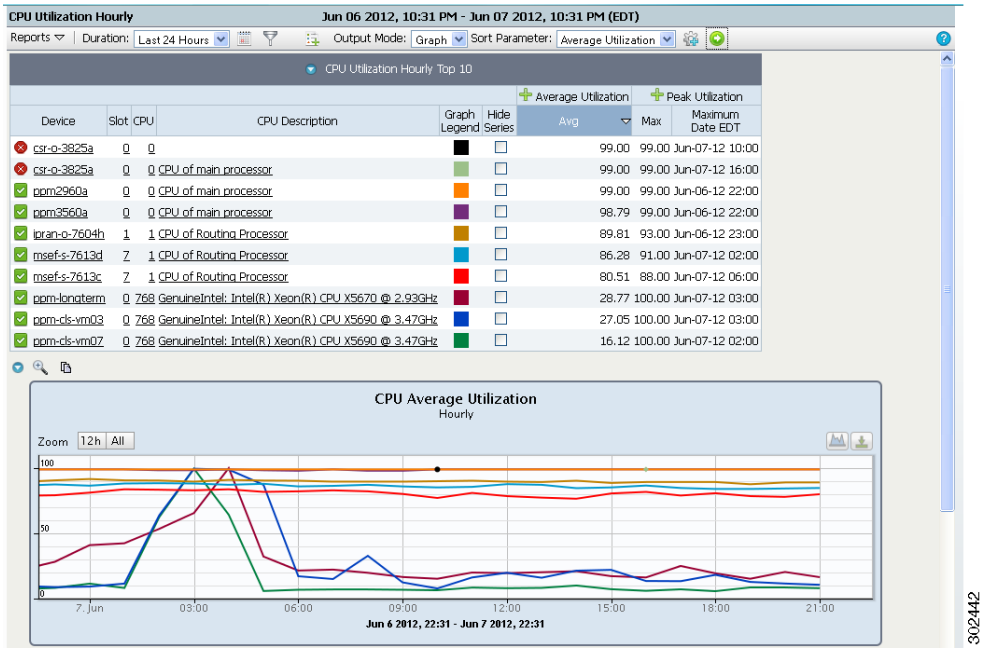
- **Graph Views**—Display a graph of performance over specified time intervals.
- **Table Views**—Display performance in a tabular view.
- **CSV File View**—Lets users save the report in a comma-separated value (CSV) file that they can view using a spreadsheet or a text editor.

The tutorial chapter of this guide ("Writing a Report") walks you through coding of the cpu.xml report, which shows CPU utilization for the network as a whole or for devices that a user selects.

### Graph View

Figure 1-1 shows an example of a Graph output view for the CPU Utilization report.

***Figure 1-1 CPU Utilization Report in Graph Output View***

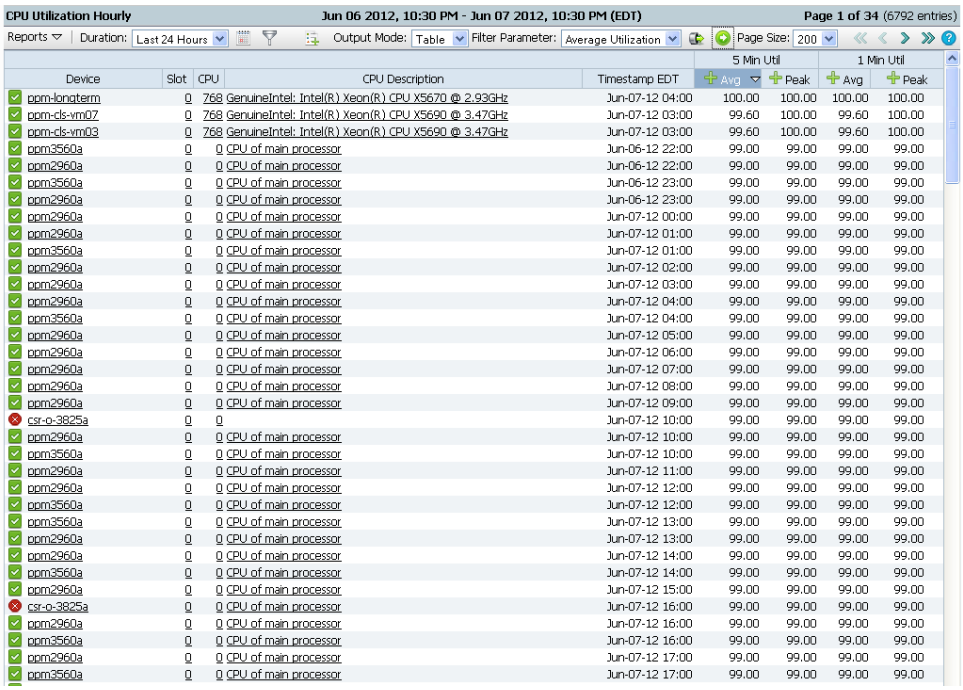



When you code your own report, you can control the time intervals that users can select.

### Table View

Figure 1-2 shows a Table output view example for the CPU Utilization report.

*Figure 1-2 Table View*




## Reporting Intervals

The XML schema for the Cisco Prime reports allows you to specify several reporting intervals for your reports.

## Sort Order

You can specify the order in which data is sorted on your reports.

## Basic Report Categories

There three basic report categories—Network Level reports, Device Level reports, and reports on specific variables. When you first view your report, it shows statistics for the entire discovered network. You can then select specific devices to view reports for a single device.

# Report Management Interface

The following report management tabs are available from the Prime Performance Manager GUI by choosing Reports from the Performance menu and any of the following:

- **Report Settings**—Allows you to enable various report intervals and control report aging.
- **Report Status**—Allows you to enable or disable reports.

- **Report Policies**—Allows you to create and manage report policies.

For details on the user interface, see the *Cisco Prime Performance Manager 1.7 User Guide*.


# Writing a Report

Writing a report for Prime Performance Manager requires careful planning and following the report writing steps in the correct order. In this chapter, we walk you through the steps for modifying a prepackaged system report—cpu.xml, and help you to develop, test, and debug your own reports.

This chapter contains:



## General Recommendations

Before you begin developing a new report, review the following recommendations:

- Complete the initial new report XML file development offline from any server until the basic report structure is ready. Then deploy the new report to the staging server for initial testing and debugging.
- Avoid developing reports on a live system. Use a staging system to develop new reports. Do not deploy new reports to your production system until they are verified to be working correctly.

## Summary of Steps for Writing a Report

1. Copy an existing report from the */opt/CSCOppm-gw/etc/pollers/system* directory to the */opt/CSCOppm-gw/etc/pollers/user* directory.
   - If needed, modify the UserCapability.xml file for the report.
   - If you are using a new MIB, compile the MIB.

2. Code a properties file for the report.
3. Code the report.
4. Enable the report using the system GUI.
5. Test and debug your report.

# Determining Which MIBs are Required

Cisco Prime Performance Manager comes with over 600 Cisco and industry standard MIBs.

To determine if the MIB that you need for your report is available:

**Step 1** Review the list of MIBs on the gateway.

- To display a list of compiled MIBs, from the Help menu, choose **Reports** and then select **SNMP MIBs**.
- All compiled MIBs are located in the in the */opt/CSCOppm-gw/etc/mibs* directory gateway.

**Step 2** If you need to add a MIB:

a. Copy the MIB to the gateway server.

b. Compile the MIB.

See Compiling a MIB, page 2-2 for the procedure for compiling a MIB.

**Step 3** If you are modifying one of the precoded reports, make sure that there is an entry in the S*ystemCapability.xml* file that references the report. If you need to change the system capability setting for the file, make any changes in the *UserCapability.xml* file.

**Note** Do not modify the *SytemCapability.xml* file.

# Compiling a MIB

If you are developing a report that a requires a new MIB (a MIB not provided with the Cisco Prime Performance Manager distribution), then you must compile the MIB before you can use it for reports.

To compile a MIB:

**Step 1** As root user, copy the MIB to the /opt/CSCOppm-gw/etc/mibs directory.

**Step 2** Enter the following command to suspend unit-gateway file synchronization.

```
# /opt/CSCOppm-gw/bin/ppm syncunits disable
```

**Step 3** Enter the following command to compile the MIBs in the system:

```
# /opt/CSCOppm-gw/bin/ppm compilemibs
```

The **compilemibs** command creates an *snmpinfo.dat* file

**Step 4** Fix any errors discovered in the MIB or reload the compiled mibs (*snmpinfo.dat*).

**Step 5** Enter the following command to restart unit-gateway file synchronization:

```
# /opt/CSCOppm-gw/bin/ppm syncunits enable
```

You can now develop a report that references the MIB.

# Determining the Statistics to Report

To determine which key performance indicators (KPIs) to report for the devices that you are monitoring:

**Step 1** Look at the compiled MIBs.

**Step 2** Based on your reporting requirements, determine which MIB variables to poll.

**Step 3** Check whether the KPIs you want to report on are covered in existing reports.

# Creating the Report XML and Properties Files

The XML file that you create can be:

- A new file based on a copy of an existing Prime Performance Manager report.
- A modification of a user-defined report that already exists for your installation.

**Caution** Do not modify the report files or properties files in the /opt/CSCOppm-gw/etc/pollers/system directory.

To create the report files:

**Step 1** If you are modifying an existing prepackaged report, copy the *.xml* file for the report and its associated *.properties* file from the */opt/CSCOppm-gw/etc/pollers/system* directory to the */opt/CSCOppm-gw/etc/pollers/user* directory or to a working directory of your own choice.

**Step 2** Rename the copied files. Make sure that they have unique filenames that are not duplicated in the */opt/CSCOppm-gw/etc/pollers/system* directory.

**Step 3** If you are using an existing user developed file in the */user* directory, copy and rename the file and its associated *.properties* file.

**Note** If a report file exists in the ../system and ../user directories, the /user report file will have precedence.

# Coding the Report

Now you are ready to code your report. The following sections take you through the cpu.xml file. This is an existing Prime Performance Manager report file that serves as our "Hello World" example for report writing. We also describe an associated file, the cpu.properties file.

- The cpu.xml file sets up polling of a standard Cisco MIB—CISCO-PROCESS-MIB.my, which reports on active system processes. It also uses the Cisco entity MIB, CISCO-ENTITY-MIB.my.
- The cpu.properties file serves only one purpose. That is, to define variables and text strings that the system uses to display the CPU reports.

Before we start, note these points:

- Since this section is a tutorial, we start with the basics—a walkthrough of the XML elements that you'll use in your reports.
- Certain advanced topics, such as coding a UserCapability.xml file modeled on the *SystemCapability.xml file* provided on the Prime Performance Manager gateway, are covered in detail in later sections of this guide.
- Later in this chapter you will find a reference section on typical report writing tasks (See Coding the Report, page 2-4 for typical report writing tasks)—some of these tasks are also covered in the tutorial.

**Tip** While working with the sample report, you can save time by keeping the online report help for the cpu.xml file open in your browser. To see the help page for cpu.xml, go to the Cisco Prime Performance Manager reports tree, choose **CPU** to bring up the CPU report, then click the report help tool: 

Example 2-1 shows the cpu.xml file.

***Example 2-1 The cpu.xml File***

```
<?xml version="1.0"?>
<!-- Copyright (c) 2011 Cisco Systems, Inc. All rights reserved. -->

<!-- MIBS Used

     CISCO-PROCESS-MIB.my
     ENTITY-MIB.my

-->

<ns:PollerList
    xmlns:ns="http://cisco.com/ppm/poller">

    <Poll name="CPU" reportId="CPU">

    <Criteria>CISCO_PROCESS_MIB</Criteria>

    <PollDefinition>

      cpmCPUTotalTable = poll("cpmCPUTotalIndex,
                              cpmCPUTotalPhysicalIndex,
                              cpmCPUTotal5minRev,
                              cpmCPUTotal1minRev");

      cpmCPUThresholdTable = poll("cpmCPUTotalIndex,
```

```
                                    cpmCPUThresholdClass,
                                    cpmCPURisingThresholdValue,
                                    cpmCPUFallingThresholdValue");

      cpmCPUThresholdTable =
         cpmCPUThresholdTable.filter(cpmCPUThresholdClass == 1);

      cpmCPUTotalTable =
        cpmCPUTotalTable.leftJoin(cpmCPUThresholdTable,
  (cpmCPUTotalTable.cpmCPUTotalIndex == cpmCPUThresholdTable.cpmCPUTotalIndex));

    </PollDefinition>

   <ProcessPollResult>

      setCpuInfo();

      fiveMinUtil       = cpmCPUTotal5minRev / 100;
      oneMinUtil        = cpmCPUTotal1minRev / 100;
      risingThreshold   = cpmCPURisingThresholdValue / 100;
      fallingThreshold  = cpmCPUFallingThresholdValue / 100;

   </ProcessPollResult>

   <ProcessDBSummary name="CPU" baseTableName="CPU">

      <Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
      <Var name="CPUNum"      type="Integer" key="true">cpuNum</Var>
      <Var name="CPUDescr"    type="String" key="true">cpuDescr</Var>

      <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
      <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>

      <Var name="CPUUtilMax1min" type="Double" operation="Max">oneMinUtil</Var>
      <Var name="CPUUtilAvg1min" type="Double" operation="Avg">oneMinUtil</Var>

      <Var name="CPURisingThreshold" type="Double">risingThreshold</Var>
      <Var name="CPUFallingThreshold" type="Double">fallingThreshold</Var>

    </ProcessDBSummary>

  </Poll>

  <CSV name="CPU" location="gateway" listen="CPU">
    <Column name="Slot">CPUSlot</Column>
    <Column name="Number">CPUNum</Column>
    <Column name="Description">CPUDescr</Column>

    <Util   name="CPUUtilMax5min">CPUUtilMax5min</Util>
    <Util   name="CPUUtilAvg5min">CPUUtilAvg5min</Util>
    <Util   name="CPUUtilMax1min">CPUUtilMax1min</Util>
    <Util   name="CPUUtilAvg1min">CPUUtilAvg1min</Util>

    <Column name="CPURisingThreshold">CPURisingThreshold</Column>
    <Column name="CPUFallingThreshold">CPUFallingThreshold</Column>
  </CSV>

<!-- *** CPU Utilization *** -->

  <WebReport name="wrnCPUUtil"
             category="level1Resources,level2CPU"
             reportId="CPU"
             context="Network,Node,CPUSlot,CPUNum,CPUDescr"
             textProps="cpu"
```

```
            sortWeight="2">

    <Criteria>CISCO_PROCESS_MIB</Criteria>

    <GraphView>
      <GraphSummary title="gstCPUUtil" />
      <Graph title="gtCPUUtilAvg" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
      </Graph>
      <Graph title="gtCPUUtilPeak" >
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </Graph>
      <LeafGraph title="gtCPUUtil" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </LeafGraph>
    </GraphView>

    <TableView baseTable="CPU">
      <IdLabel/>
      <Label colSpan="2" name="cpuUtil5Min" />
      <Label colSpan="2" name="cpuUtil1Min" />
      <Label colSpan="2" name="cpuUtilThresh" />
      <HeaderRow/>

      <Link name="node" context="Node">fqdnid</Link>
      <Link name="slot" context="CPUSlot">CPUSlot</Link>
      <Link name="cpu"  context="CPUNum">CPUNum</Link>
      <Link name="cpuDes"  context="CPUDescr">CPUDescr</Link>
      <Time/>


      <Util default="true" name="avg">    CPUUtilAvg5min</Util>
      <Util                name="peak">   CPUUtilMax5min</Util>

      <Util                name="avg">    CPUUtilAvg1min</Util>
      <Util                name="peak">   CPUUtilMax1min</Util>

      <Util                name="rising"> CPURisingThreshold</Util>
      <Util                name="falling">CPUFallingThreshold</Util>
    </TableView>

  </WebReport>

</ns:PollerList>
```

# Main Elements of Report XML

The cpu.xml report has seven main sections:

- **Comments**— The top of the cpu.xml file contains a comment that identifies the MIBs polled. However, you can place comments anywhere in the XML file.

All of the report sections are specified within one XML section called PollerList. The first line of the PollerList section specifies “http://cisco.com/ppm/poller.” Always include this as stated in the sample reports. You can provide more than one Poll section.

The Poll section is the main element of the XML file, and encapsulates four sections.

Poll is a complex element type that contains elements defined in the *EventPoller.xsd* schema file, within the Processor element:

- **Criteria Section**—Specifies a capability value that is defined in the *SystemCapability.xml* file or the *UserCapability.xml* file.

  See Specify the MIB Criteria (Criteria Section), page 2-8.

- **PollDefinition Section**—Calls report macros that create a virtual table where polling results are stored.

  See Specify the Poll Name and Report Name (Poll Element), page 2-8.

- **ProcessPollResult Section**—Calls reporting macros and performs operations that modify or format the polling data using formulas or other methods.

  See Specify Poll Processing Results (ProcessPollResult Section), page 2-10.

- **ProcessDBSummary Section**—Sets up and manipulates the table and row format for a virtual table that holds the polled data for the report. At the end of each defined processing interval, the system writes the table data for that processing interval to the physical database on the gateway.

  See Assign the Data to the Database Schema (ProcessDBSummary Section), page 2-11.

The final two sections specify the appearance and format of the user reports.

- **CSV**—Specifies the layout of the CSV file that users create when they view the file in the opt/CSCOppm-gw/reports directory or when they use the export to CSV option on every web report.

  See Specify the CSV Output File and Format (CSV Section), page 2-12.

- **WebReport**—Specifies the attributes of the graphical report that users see when they select the Graph View option. The web report also specifies what is shown in the table view and how it is shown. The WebReport section contains:

  - **Attributes**—These specify the name of the report in Prime Performance Manager menus, where in the report tree the report appears, where the report data comes from, the Properties file for the web report, and so on.

    See Specifying the Attributes for the WebReport Section, page 2-14.

  - **GraphView**—Specifies the attributes of the graph view that users see when they select the Graph View option.

    See Coding the GraphView Section, page 2-15.

  - **TableView**—Specifies the attributes of the table view that users see when they select the Table view option.

    See Coding the TableView Section, page 2-18.

**Note** Web reports can also be created and edited using the Prime Performance Manager GUI.

# Report Macros

You can use the report macros provided with the Prime Performance Manager gateway in your XML code. Some of the report macros can be used only in specific sections of the XML. For reference information on the report macros, see Chapter B, "Reports Macro Reference."

# Add a Comment Stating the MIBs Used for the Report

At the top of your report, include a comment that identifies the MIBs polled for your report. The cpu.xml file contains the following comments:

```
<!-- MIBS Used

     CISCO-PROCESS-MIB.my
     ENTITY-MIB.my

-->
```

The extension of the MIBs used should have a .my or .mib suffix.

# Specify the MIB Criteria (Criteria Section)

This section deals with an advanced topic in report coding: specifying the criteria used to process the MIBs for your report.

The *SystemCapability.xml*l specifies the default MIB processing criteria for Prime Performance Manager reports. Do not edit the *SystemCapability.xml* file. If you need to modify MIB processing criteria, edit the UserCapability.xml file and specify your processing criteria there.

To add a new capability or criteria, add an entry similar to the following to UserCapability.xml:

```
CISCO_PROCESS_MIB = isTableNotEmpty("cpmCPUTotalTable");
```

This entry will check if the MIB table cpmCPUTotalTable on the device has data and set the CISCO_PROCESS_MIB capability if it does. Or you can add:

```
UDP_MIB  = hasVar("udpInDatagrams");
```

This entry sets the UDP_MIB capability if the MIB variable udpInDatagrams exists on the device. Macros available for this include isTableNotEmpty() and hasVar(). For information, see Chapter B, "Reports Macro Reference."

# Specify the Poll Name and Report Name (Poll Element)

The Poll element:

- Specifies the name of the poller and the Report ID. You can use multiple reportIDs for reports under same poll but categorized to different categories, for example:

  ```
  name=Interface
  reportId=INTERFACE,INTERFACEINTRA,INTERFACEINTER,INTERFACEERRORS
  ```

  Make sure to assign a unique name for the poller and the report ID. Also, if you give the different levels of report hierarchy in the web report section for the same poll section, assign a unique report ID.

  **Note** The values for name and reportId are case sensitive.

  For example, if you copy the cpu.xml file to another XML file in your /user directory, give the poller and the report ID a new name that is not duplicated in the */opt/CSCOppm-gw/etc/pollers/system* directory or the */opt/CSCOppm-gw/etc/pollers/user* directory.

- Is a complex XML element that contains the next main XML sections that set up polling and processing of polling results.

# Specify the MIB Variables to Poll (PollDefinition Section)

The PollDefinition section can be used to poll using SNMP (poll), CSV (csvPoll) or CLI (xmlPoll). For SNMP, the MIB parameters are specified in this section. The section defines the polling that is performed for the report and assigns the polled data to internal database tables that Prime Performance Manager uses to generate the reports.

Example 2-2 shows the PollDefinition section for the sample cpu.xml report.

***Example 2-2 PollDefinition Section***

```
<PollDefinition>

      cpmCPUTotalTable = poll("cpmCPUTotalIndex,
                               cpmCPUTotalPhysicalIndex,
                               cpmCPUTotal5minRev,
                               cpmCPUTotal1minRev");

      cpmCPUThresholdTable = poll("cpmCPUTotalIndex,
                                   cpmCPUThresholdClass,
                                   cpmCPURisingThresholdValue,
                                   cpmCPUFallingThresholdValue");

      cpmCPUThresholdTable =
         cpmCPUThresholdTable.filter(cpmCPUThresholdClass == 1);

      cpmCPUTotalTable =
        cpmCPUTotalTable.leftJoin(cpmCPUThresholdTable,
  (cpmCPUTotalTable.cpmCPUTotalIndex == cpmCPUThresholdTable.cpmCPUTotalIndex));

    </PollDefinition>
```

This section polls specific variable in the *CISCO-PROCESS-MIB.my* and stores them in two tables:

- cpmCPUTotalTable
- cpmCPUThresholdTable.

The MIB objects are standard objects in the CISCO-PROCESS-MIB that provide CPU load monitoring. For additional information and comments, refer to the *CISCO-PROCESS-MIB.my* file.

### The POLL Macro

The actual polling is done using the POLL macro. This macro has the following syntax:

POLL(*arg1*, *arg2 ... argn*)

where the arguments are a list of MIB variables to be polled, enclosed in quotes.

### The FILTER Macro

Next, the PollDefinition section calls a macro that is defined for PollDefinition objects, the FILTER macro:

```
cpmCPUThresholdTable =
         cpmCPUThresholdTable.filter(cpmCPUThresholdClass == 1);
```

The FILTER macro has the following syntax:

FILTER(*object*, *arg1*)

Where *object* is the table object passed to the macro and *arg1* specifies the filtering criterion. Here we are filtering on the cpmCPUThresholdClass. CPU data for objects in the table that have a cpmCPUThresholdClass value of 1 is retained, and objects and their data that do not match are removed from the table.

Now we have two tables: a cpmCPUTotalTable and a cpmCPUThresholdTable. To maximize processing efficiency, we want to combine these two tables into one virtual table. The next line in the PollDefinition section does that:

```
cpmCPUTotalTable =
        cpmCPUTotalTable.leftJoin(cpmCPUThresholdTable,
  (cpmCPUTotalTable.cpmCPUTotalIndex == cpmCPUThresholdTable.cpmCPUTotalIndex))
```

This is done by calling the LEFTJOIN macro. This macro looks at the cpmCPUThresholdTable and the cpmCPUTotalTable, and if the index values are the same, joins rows from the two tables into one table.

Now we have one table that contains all of the data that we want to process.

#### The LEFTJOIN Macro

The LEFTJOIN macro has the following syntax:

LEFTJOIN (*object*, *arg1*, *arg2*)

where *object* and *arg1* are tables and *arg2* is the condition whether each row has a match.

The macro returns the resulting joined tables of *object* and *arg1*.

A row from *object* and a row from *arg1* are joined together if the condition (*arg2*) is true. However, each object row will be retained in the resulting table even if it does not match any row from the object specified in *arg1*.

**Note** setCpuInfo() is a special macro that gets data from the ENTITY-MIB and fills in the data for the mentioned fields.

## Specify Poll Processing Results (ProcessPollResult Section)

The ProcessPollResult section sets up variables to manipulate the data and processes it to obtain the information for the CPU report.

Example 2-3 shows the ProcessPollResult section of the cpu.xml file.

***Example 2-3 ProcessPollResult Section***

```
<ProcessPollResult>

      setCpuInfo();

      fiveMinUtil       = cpmCPUTotal5minRev / 100;
      oneMinUtil        = cpmCPUTotal1minRev / 100;
      risingThreshold   = cpmCPURisingThresholdValue / 100;
      fallingThreshold  = cpmCPUFallingThresholdValue / 100;

   </ProcessPollResult>
```

This is what we need to do:

1. We want to calculate percentage values for CPU average utlilization over two of the time intervals defined in the CISCO-PROCESS-MIB: `cpmCPUTotal5minRev` and `cpmCPUTotal1minRev` (5 minute intervals and 1 minute intervals).
2. We also want to calculate percentage values for CPU average peak utilization. The data used to calculate this number is in the `cpmCPURisingThresholdValue` and `cpmCPUFallingThresholdValue` values.
3. We start by calling the SETCPUINFO macro. In the absence of a specified index, this macro simply sets the CPU description, CPU number, and CPU slot for each CPU whose data is returned in the table.

The percentage figures used to determine CPU average utilization and CPU average peak utilization are calculated by dividing the data for each MIB variable by 100.

We now have the data that we want to display in our Prime Performance Manager reports.

Now we want to assign the calculated values to the Prime Performance Manager gateway's database. This is done in the next section of the XML code—the ProcessDBSummary section.

# Assign the Data to the Database Schema (ProcessDBSummary Section)

The ProcessDBSummary section sets up table columns in virtual data tables. The var order is not important. Vars are looked up by name when reports are run.

Example 2-4 shows the ProcessDBSummary section for the cpu.xml report.

***Example 2-4 ProcessDBSummary Section***

```
<ProcessDBSummary name="CPU" baseTableName="CPU">

      <Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
      <Var name="CPUNum"      type="Integer" key="true">cpuNum</Var>
      <Var name="CPUDescr"    type="String" key="true">cpuDescr</Var>

      <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
      <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>

      <Var name="CPUUtilMax1min" type="Double" operation="Max">oneMinUtil</Var>
      <Var name="CPUUtilAvg1min" type="Double" operation="Avg">oneMinUtil</Var>

      <Var name="CPURisingThreshold" type="Double">risingThreshold</Var>
      <Var name="CPUFallingThreshold" type="Double">fallingThreshold</Var>

    </ProcessDBSummary>
```

The first line of the ProcessDBSummary section specifies the name of the poll and the name of the report.

The rest of the code in the ProcessDBSummary section basically sets up the columns that will be set up in each row of the virtual database.

The operations specified in the operation variables for some of the table columns are defined in the *EventPoller.xsd* schema file.

The code sets up the table columns in variables that set up the columns. For example, the first three lines of the section set up the CPU slot number, CPU number, and CPU description.

```
<Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
```

```
<Var name="CPUNum"      type="Integer" key="true">cpuNum</Var>
<Var name="CPUDescr"    type="String" key="true">cpuDescr</Var>
```

Two of the columns used for the Average Utilization and Peak Utilization parts of the report require calculation of an average value and a maximum value. These are specified by the lines that contain `operation="Max"` or `operation = "Avg"` as shown below:

```
<Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
<Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>

<Var name="CPUUtilMax1min" type="Double" operation="Max">oneMinUtil</Var>
<Var name="CPUUtilAvg1min" type="Double" operation="Avg">oneMinUtil</Var>
```

At the end of each defined polling interval, the data for the virtual table set up for the specified polling interval is written to the physical database on the Prime Performance Manager gateway.

# Setting Up Equivalencies

The Equivalence section appears in the PollerList, not in the Poll section like ProcessDBSummary. The baseTableName is the name of the table that contains the variables that you can use to logically link table data. For example, if you want the variable ifDescr, from the INTERFACE table, to be interchangeable with the inputIf variable in NETFLOW_ALLFLOWS, NETFLOW_SOURCES you can do the following:

```
<Equivalence>
    <EquivalenceKey baseTableName="INTERFACE" variableName="ifDescr"/>
    <EquivalenceKey baseTableName="NETFLOW_ALLFLOWS" variableName="inputIf"/>
    <EquivalenceKey baseTableName="NETFLOW_SOURCES" variableName="inputIf"/>
</Equivalence>
```

With this Equivalence section in place, users can go from an Interface level Transport Statistics -> Interface report to any NetFlow report using the ALLFLOWS or SOURCES baseTable. When the NetFlow report is run, the inputIf is replaced with the value of the ifDescr of the interface you were looking at on the Interface report.

At this point, we have extracted the data from the variables polled for the MIB, manipulated the data, and set up the tables for the reporting on each polled device.

Now we are ready to specify how the reports look. This is done in the next two sections Specify the CSV Output File and Format (CSV Section), page 2-12 and Set up the Web Reports (GraphReport and TableView Sections), page 2-13.

# Specify the CSV Output File and Format (CSV Section)

The CSV section specifies the layout of the data that is written to a comma separated value (CSV) file when users select the CSV report option. The name tag specifies the filename prefix for all files generated for this report.

Example 2-5 shows the CSV section of the cpu.xml report.

***Example 2-5 CSV Section***

```
<CSV name="CPU" location="gateway" listen="CPU">
    <Column name="Slot">CPUSlot</Column>
    <Column name="Number">CPUNum</Column>
    <Column name="Description">CPUDescr</Column>
```

```
    <Util   name="CPUUtilMax5min">CPUUtilMax5min</Util>
    <Util   name="CPUUtilAvg5min">CPUUtilAvg5min</Util>
    <Util   name="CPUUtilMax1min">CPUUtilMax1min</Util>
    <Util   name="CPUUtilAvg1min">CPUUtilAvg1min</Util>

    <Column name="CPURisingThreshold">CPURisingThreshold</Column>
    <Column name="CPUFallingThreshold">CPUFallingThreshold</Column>
  </CSV>
```

The first line of the CSV section specifies the following values:

- The name of the CSV file.
- The location of the report, in this case the Prime Performance Manager gateway.
- The listen variable, which base table name is being used.

The following attributes can be defined if you want the CSV section to be independently enabled or disabled, for example you want a CSV report generated but not display the report in the GUI. These attributes are also in the WebReport section and have the same purpose:

- reportId is a unique identifier that allows you to specify which set of reports to enable/disable together. Everything with the same reportId are toggled together.
- category is the category the report will appear under on the Report Status page where it can be enabled or disabled. If category is not defined in the WebReport section, this category will not appear in the report tree on the left hand side of the GUI. This allows you to create CSV reports that aren't displayed in the GUI.
- textProps is optional and lets you point to a properties file in the report directory to look up human readable names. If it isn't set, it will pick up the name corresponding to the XML file that contains it. For example, if this was placed in interface.xml, it uses interface as the textProps and looks for interface.properties.

If you add them to the CSV section, reportId and category are required; textProps is optional.

The remaining lines in the CSV section simply specify the text string or database column data.

```
Column name="Slot">CPUSlot</Column>
    <Column name="Number">CPUNum</Column>
    <Column name="Description">CPUDescr</Column>

    <Util   name="CPUUtilMax5min">CPUUtilMax5min</Util>
    <Util   name="CPUUtilAvg5min">CPUUtilAvg5min</Util>
    <Util   name="CPUUtilMax1min">CPUUtilMax1min</Util>
    <Util   name="CPUUtilAvg1min">CPUUtilAvg1min</Util>

    <Column name="CPURisingThreshold">CPURisingThreshold</Column>
    <Column name="CPUFallingThreshold">CPUFallingThreshold</Column>
```

Now we are ready to set up the web reports.

# Set up the Web Reports (GraphReport and TableView Sections)

The WebReport section (or XML element) includes:

- **Attributes**—Specify the name of the report in Prime Performance Manager menus, where in the report tree the report appears, where the report data comes from, the Properties file for the web report, and so on
- **GraphView Section**—Sets up the graph view for the report.

- **TableView Section**—Sets up the table view for the report.

## Specifying the Attributes for the WebReport Section

The attributes for the WebReport section specify key information that determines where the report is displayed and what data it uses.

The attributes set values that are specified in the Properties file for the XML report. For more details on the Properties file for the cpu.xml report, see Specifying the Attributes for the WebReport Section, page 2-14.

Example 2-6 shows the attributes specified for the WebReport section of the cpu.xml file.

***Example 2-6 WebReport Attributes***

```
<WebReport name="wrnCPUUtil"
             category="level1Resources,level2CPU"
             reportId="CPU"
             context="Network,Node,CPUSlot,CPUNum,CPUDescr"
             textProps="cpu"
             sortWeight="2">
```

The code shown in Example 2-6 sets these attributes:

- **name**—Specifies a unique name, which is specified in the Properties file (cpu.properties). This is the report name that appears in Prime Performance Manager menus. The name does not have to map to a properties file. A value given here that does not map to a properties file is shown as-is.
- **category**—Specifies where the report appears in the Prime Performance Manager report tree.

  For example, category=level1Resources,level2CPU specifies that the CPU report will appear under the Resources report category, and will be identified by the text, CPU. The relationships between level1Resources and Resources and level2CPU and CPU come from the properties file. If a properties file does not exist, a report with this category would literally be in the level1Resources category and the level2CPU sub-category.
- **reportId**—Needs to map to a reportId defined in the Poll definition.
- **context**—Specifies the drill-down options available with the report and where the report can appear.
- **textProps**—Specifies the name of the Properties file to use: cpu.properties.
- **sortWeight**—Specifies the order of the report within the category to which it is assigned. It is one of many optional properties defined in the schema.

  The line sortWeight=2 specifies order in which the report types appear in the Reports list.

## How the Properties File is Used

Properties files have only one purpose: to set up variables to hold text strings displayed in the web GUI when users select the Graph View or the Table View.

**Note** CSV names are given within the CSV column/util tag in the XML and not in the properties file.

Example 2-7 shows the properties file for the cpu.xml report.

***Example 2-7 cpu.properties File***

```
level1Resources = Resources
level2CPU       = CPU

#CPU Utilization
wrnCPUUtil      = CPU Utilization
gstCPUUtil      = CPU Utilization

gtCPUUtilAvg     = CPU Average Utilization

genCPUUtilAvg    = Average Utilization

gtCPUUtilPeak    = CPU Average Peak Utilization

gtCPUUtil        = CPU Average and Peak Utilization

genCPUUtilPeak   = Peak Utilization

avg           = Avg
peak          = Peak

rising        = Rising
falling       = Falling

node          = Node
slot          = Slot
cpu           = CPU
cpuDes        = CPU Description

cpuUtil5Min   = 5 Min Util
cpuUtil1Min   = 1 Min Util
cpuUtilThresh = Threshold
```

## Coding the GraphView Section

The GraphView section sets up the report graph view, including:

- The graph summary table title.
- The graph title and the column names for the graph to be plotted. One graph view can have any number of graphs. The graph column may be a KPI.
- The leaf graph, if applicable.
- Column is also available. Column has several subclasses including Util, Bits, and Bytes. Column displays the value as is. Util multiplies the value by 100 and displays it as a percent.

  You can apply attributes to a column including filterable, sortable, compoundname, showPercentageColumn, and others. Showpercentagecolumn, for example, adds all the values in a column and then populates each row with the percentages of the total value for each entry. Using Disk Space Statistics as an example, you can add the showPercentageColumn attribute as follows:

1. Open the .xml file associated with the web report (disk.xml).
2. Scroll to the web report definition you want to edit (WebReport name=wrnDiskStats).
3. For at least one graph view column, add a showPercentageColumn attribute with value = true. Replace the following text:

```
<Graph title="gtPercentSpace">
<Util name="genPercentSpace">percentSpace</Util>
</Graph>
```

With this:

```
<Graph title="gtPercentSpace">
<Util name="genPercentSpace" showPercentageColumn="true">percentSpace</Util>
</Graph>
```

You should specify the text that is displayed using variables defined in the properties file for the report, as shown in Example 2-8. You can use the Bytes tag to display the byte values in Kb, Mb, and Gb format, if needed.

Example 2-8 shows the GraphView section of the cpu.xml report.

***Example 2-8 GraphView Section***

```
<GraphView>
      <GraphSummary title="gstCPUUtil" />
      <Graph title="gtCPUUtilAvg" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
      </Graph>
      <Graph title="gtCPUUtilPeak" >
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </Graph>
      <LeafGraph title="gtCPUUtil" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </LeafGraph>
    </GraphView>
```

The GraphSummary element specifies the title that appears above the report graph in the Prime Performance Manager GUI:

```
<GraphSummary title=”gstCPUUtil” />
```

This specifies that the title is as specified by the *gstCPUUtil* variable in the properties file (“CPU Utilization”).

#### Hiding the Summary Tables on First Report Display

If you want to hide the summary tables the first time the report is displayed, you can use the minimized attribute as in the following example:

```
<GraphSummary title="ifDashboard" minimized="true"/>
```

**Note** You can override display of the summary titles on a per-user basis.

## Coding the Graph Sections

The Graph sections within the GraphView section specify the text strings that will identify the two graphs shown in the graph view for the CPU report, as defined in the properties file. These are, CPU Average Utilization and CPU Average Peak Utilization.

Within the Graph section, you can:

- Simply display values “as is” from the database, by coding a Column element. A column element (or util) can also perform operations. For example:

  ```
  (Column name="someName") (CPUUtilAvg5min + CPUUtilMax5min)/2 (/Column)
  ```

- Perform an operation on the data, by coding a Util element.

The code in the Graph sections for the cpu.xml report specifies Util values that multiply the base values by 100.

## Merging Graphs in Dashboards

Users can use the Prime Performance Manager GUI to merge graphs in views. While users cannot merge graphs in dashboards using the GUI, you can merge graphs in a dashboard using the syntax provided in the following example. The example uses the third table and graph in Resource Dashboards > CPU 1Min/5Min/Memory Stats. The objective is to create a table with used bytes, free bytes, and utilization shown together. The example also creates a graph with used and free bytes displayed together. (Utilization is not included because it is a percentage and the bytes are raw values.)

The following format, from cpuMemDash.xml, performs the table graph merge:

```
<GraphView title="gtMemPoolBytes" graphsToMerge="2">
  <GraphSummary title="gstMemPoolUtilization" />
  <Graph title="gtMemoryPoolUsedBytes" >
     <Bytes name="genMemoryPoolUsedBytes">MemoryPoolUsedBytes</Bytes>
  </Graph>
  <Graph title="gtMemoryPoolFreeBytes" >
     <Bytes name="genMemoryPoolFreeBytes">MemoryPoolFreeBytes</Bytes>
  </Graph>
  <Graph title="gtMemPoolUtilAvg" showCurrent="true" hideMax="true">
     <Util name="genMemPoolUtilAvg">MemoryPoolUtilAvg</Util>
  </Graph>
  <LeafGraph title="gtMemPoolUtil">
     <Util name="genMemPoolUtilAvg">MemoryPoolUtilAvg</Util>
     <Util name="genMemPoolUtilMax">MemoryPoolUtilMax</Util>
  </LeafGraph>
</GraphView>
```

The report has three graphs: used bytes, free bytes, and utilization. These correspond to the three data columns displayed in the summary table. In the <GraphView> tag, an optional graphsToMerge attribute indicates the first two graphs inside the graph view are to be merged. If you do not specify graphsToMerge, all graphs are merged.

## Coding a LeafGraph Section

If your report drills down through the available subreports until the lowest level is reached, and only one instance of an object you are reporting on is left, you can code a LeafGraph section to combine the different data metrics for that object in one graph.

For example, for the CPU report, users can click on the text in the CPU Description column for each processor to display a combined report called the CPU Average and Peak Utilization report. Such sub-reports are defined in a LeafGraph section.

The LeafGraph section for the CPU report is specified as follows:

```
<LeafGraph title="gtCPUUtil" >
        <Util name="genCPUUtilAvg">CPUUtilAvg5min</Util>
        <Util name="genCPUUtilPeak">CPUUtilMax5min</Util>
      </LeafGraph>
```

The title attribute for the LeafGraph section specifies a variable from the properties file (gtCPUUtil) that specifies the title of the Leaf report: CPU Average and Peak Utilization. Lines are the default chart data series output style. You can specify the basic bar or stacked percentage column output styles by adding the type attribute, as follows:

```
<LeafGraph title="gtCPUUtil" type="bar">
```

or,

```
<LeafGraph title="gtCPUUtil" type="StackedPercentageColumn">
```

Users can modify chart output styles individually. For information, see the "Managing Reports, Dashboards, and Views" chapter the the *Cisco Prime Performance Manager 1.7 User Guide*.

The Util section specify the titles of two reports items shown in a table at the top of the leaf report Peak Utilization and Average Utilization, and the Util variables specify the table data that is displayed.

## Coding the TableView Section

The final section of the cpu.xml report sets the format for the table view for the report.

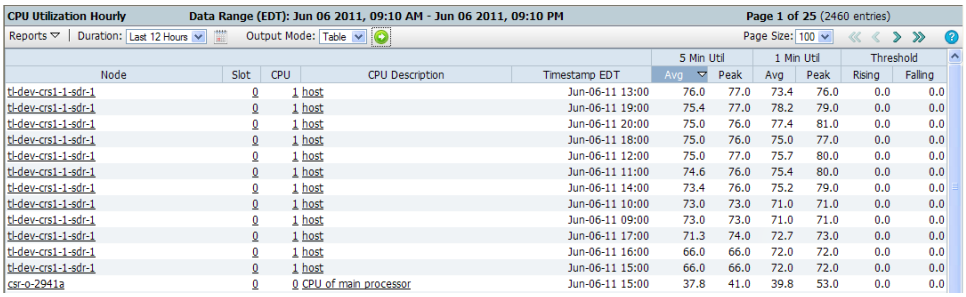
Example 2-9 shows the TableView section for the cpu.xml report.

***Example 2-9 TableView Section***

```
<TableView baseTable="CPU">
      <IdLabel/>
      <Label colSpan="2" name="cpuUtil5Min" />
      <Label colSpan="2" name="cpuUtil1Min" />
      <Label colSpan="2" name="cpuUtilThresh" />
      <HeaderRow/>

      <Link name="node" context="Node">fqdnid</Link>
      <Link name="slot" context="CPUSlot">CPUSlot</Link>
      <Link name="cpu"  context="CPUNum">CPUNum</Link>
      <Link name="cpuDes"  context="CPUDescr">CPUDescr</Link>
      <Time/>


      <Util default="true" name="avg">    CPUUtilAvg5min</Util>
      <Util                name="peak">   CPUUtilMax5min</Util>

      <Util                name="avg">    CPUUtilAvg1min</Util>
      <Util                name="peak">   CPUUtilMax1min</Util>

      <Util                name="rising"> CPURisingThreshold</Util>
      <Util                name="falling">CPUFallingThreshold</Util>
    </TableView>
```

Figure 2-1 shows a Table View for the CPU report.

***Figure 2-1 Example of a Table View***




The code in the TableView section specifies the attributes that set up the layout of the table:

- As Figure 2-1 shows, there are several headings that have two subheadings. For example, 5 Min Util has subheadings for Avg and Peak, and Threshold has subheadings for Rising and Falling.

These are specified in the three Label sections:

```
<Label colSpan="2" name="cpuUtil5Min" />
<Label colSpan="2" name="cpuUtil1Min" />
<Label colSpan="2" name="cpuUtilThresh" />
```

The *colSpans* attribute specifies that the main heading spans two subcolumns, and the names attribute specifies the variables that define the text for the headings, as specified in the properties file. For example, cpuUtil5Min specifies the string 5 Min Util.

- Several of the report columns contain links to other items. For example, the CPU Description column contains links that display reports for the selected CPU, and the CPU Description column contains links that display reports for the selected node.

  The Link sections define these links, as follows:

  ```
  <Link name="node" context="Node">fqdnid</Link>
  <Link name="slot" context="CPUSlot">CPUSlot</Link>
  <Link name="cpu"  context="CPUNum">CPUNum</Link>
  ```

  The name attribute in each Link section specifies a string, defined in a variable in the properties file, that describes the item; for example slot specifies “Slot.” The context attribute specifies a column in the ProcessDBSummary section, for example, for slot, the data column for CPUSlot is displayed.

  The Time section specifies the columns from the DBSummary table that contain data for each time range report and the heading for the subcolumn. For example for the Rising subcolumn for the Threshold column, the code specifies a Util value:

  ```
  <Util name="rising"> CPURisingThreshold</Util>
  ```

  If you don’t want to apply a Util operation to the data, you can code a Column element instead—this will display the data unchanged.

## Creating a Report with Thresholdable Fields

Many Prime Performance Manager table and graph view fields are prefixed with a + to indicate users can create thresholds using the Prime Performance Manager Threshold Editor. (For information on creating thresholds, see “Managing Thresholds” in the *Cisco Prime Performance Manager 1.7 User Guide*.)

Any numeric column value defined in the report XML file WebReport declarations can be configured as a threshold as long as the value is not a key or link value and does not have the 'thresholdable="false"' attribute.

When users create thresholds, they edit the key performance indicator (KPI) properties. The KPI can be either rising or falling. For a rising threshold, the critical alarm threshold must be higher than the major alarm threshold, and the major alarm threshold must be higher than the minor alarm. For a falling threshold, the critical alarm entry must be lower than the major alarm, and the major alarm must be lower than the minor alarm.

You can use the tcaRising attribute to create a thresholdable field with the expected rising or falling KPI value. This attribute is placed under the GraphView or TableView declaration in the report XML file. If you set tcaRising=true (default), the field’s KPI will be rising. If you set tcaRising=false, the KPI will be falling.

Fields for which you cannot use the tcaRising attribute include:

- Any LeafGraph fields.
- Any Dashboard report XML file field. Dashboard reports have *Dash.xml file names.

Following is an example of a thresholdable field based on ping.xml

```
<!-- *** ICMP Ping Availability *** -->

  <WebReport name="wrnICMPPingAvail"
             category="level1Availability,level2ICMPPing"
             reportId="ICMP_PING"
             context="Network,Node,"
             sortWeight="22"
             textProps="ping">

    <GraphView>

       <GraphSummary title="gstICMPPingAvail" />
       <Graph title="gtICMPPingAvailAvg" >
          <Util name="genICMPPingAvailAvg" default="true"
                descending="false" tcaRising="false">PingAvailability</Util>
       </Graph>

       <LeafGraph title="gtICMPPingAvailAvg" type="bar">
          <Util name="genICMPPingAvailAvg">PingAvailability</Util>
NOTE: the tcaRising attribute is not added here.
       </LeafGraph>

    </GraphView>

    <TableView baseTable="ICMP_PING">
      <IdLabel/>
      <Label colSpan="1" name="icmpPingAvail" />
      <Label colSpan="3" name="latency" />
      <HeaderRow/>

      <Link name="node"  context="Node">fqdnid</Link>
      <Time/>

      <Util name="genICMPPingAvailAvg" default="true"
            descending="false" tcaRising="false">PingAvailability</Util>

      <Column name="avg">PingLatencyAvg</Column>
      <Column name="max">PingLatencyMax</Column>
      <Column name="min">PingLatencyMin</Column>

    </TableView>
```

# Providing an Online Help File

After you develop your report, you can provide an online help file for it. If you provide customized Online help for your report, it appears when users choose **Custom Help** on the Help page for reports.

To create your own online help file:

**Step 1** Create an HTML file that contains the title of your report. For example, if you report is named *test.xml*, create an HTML file named *test.xml.custhlp*.html.

**Step 2** Write the content for the file.

**Step 3** Copy the help file to the following directory on the gateway:

*/opt/CSCOppm-gw/apache/share/htdocs/reportHelp/user/*

## Manually Generating the Online Help Files

If you want to manually generate the system-generated online help files, enter the following from the gateway command line:

```
ppm docreps
```

# Best Practices

Best practices to follow when you develop reports include:

- Do not edit files in the /system directory.

  Do not edit the files in the */opt/CSCOppm-gw/etc/pollers/system* directory. If you want to use an existing report file and its associated property file as the starting point for a new customized report, copy the files to the */opt/CSCOppm-gw/etc/pollers/user* directory and work on them there. If the file has same name, the user file is used. This is best way to create a customized version of an existing system file. You can also pen the report using the Web Report Editor GUI, which will save a copy under the user directory.

- Do not edit the SystemCapability.xml file.

  Do not edit the SystemCapability.xml file, which is located in the */opt/CSCOppm-gw/etc* directory. To modify the capabilities used for your own reports, edit the *UserCapability.xml* file as required. This file is also located in the */opt/CSCOppm-gw/etc* directory.

- Add new .xml and .properties files in the user directory.

  If you need to add new *.xml* files or *.properties* files, add these files and edit them in the */opt/CSCOppm-gw/etc/pollers/user* directory.

- Be careful when modifying the schema used in previous reports.

  As you develop the code for your report, exercise care when modifying the schema that is used for previously developed reports. If you modify the internal schema without considering the previous coding that affects it, this might cause the internal database to become unstable and require reinitializing the database.

- Use unique poller names, report IDs, and database table names for your reports.

  Make sure that you always use unique poller names, report IDs and database table names for your reports. If you use an existing name, this can cause serious system issues.

- Use a properties file to specify common settings.

  Use the *.properties* file associated with your report's XML file to specify settings that are used in more than one section of your reports or used across several reports. This ensures consistent processing and helps eliminate errors that can result from coding the same variables more than once.

# Task Reference

This section provides reference information for common tasks you might perform while developing reports. Tasks include:

- Enabling 1-Minute and 5-Minute Reports, page 2-22
- Adding a New Column to an Existing Report, page 2-22

# Enabling 1-Minute and 5-Minute Reports

1-minute and 5-minute reports are not enabled by default. You can enable 1-minute and 5-minute reports by editing the XML report definition file that corresponds to the report, or you can allow users to enable1-minute and 5-minute reports using the Prime Performance Manager GUI. The ProcessDBSummary section and the WebReport section require the addition of interval attributes.

**Note** Setting the interval in the report xml is rarely needed. Report intervals are normallly controlled from the GUI report status. You can set it in the report XML if you want to restrict the options available, usually to not allow a report to set low level values such as 1 min or 5 min if the report can generate very large tables.

Here is an example of adding the interval attribute to the CPU Utilization report to enable 5-minute reporting in the cpu.xml report definition file:

```
<ProcessDBSummary name="CPU" baseTableName="CPU"
interval="Min1Min5Min15HourlyDailyWeeklyMonthly">
<WebReport name="wrnCPUUtil"
             category="wrcVendor,wrcCategory"
             reportId="CPU"
             context="Network,Node,CPUSlot,CPUNum,CPUDescr"
             textProps="cpu"
             sortWeight="2" interval="Min5Min15HourlyDaily">
```

The 1-minute and 5-minute intervals are included because of the Min1Min5 piece of the interval attribute. The 1-min and 5-minute interval is defined as

```
interval="1Min5Min15MinHourlyDailyWeeklyMonthly
```

This attribute is not defined by default and implicitly defaults to 15-minute, hourly, and daily reports.

1-minute and 5-minute reports can also be enabled globally through the GUI:

- In the Reports/Group Settings tab, enable 1-minute and 5-minute reports globally.
- In the Reports/Group Status tab, enable 1-minute and 5-minute reports for specific reports. This can also be performed at the device level.

**Note** Enabling 1-minute and 5-minute reports will significantly increase the amount of resources required for your units.

# Adding a New Column to an Existing Report

To add new columns to an existing report:

**Step 1** If the required MIB attributes are not polled in the PollDefinition section, modify the polling definitions.

**Step 2** Include the formulas required as shown in Modifying the ProcessPollResult Section, page 2-24.

**Step 3** Add the variables in the database table using the *Var* element in the <ProcessDBSummary> element (see Modifying the ProcessDBSummary Section, page 2-23).

**Step 4** Save the changes and restart the gateway and unit.

# Modifying the ProcessDBSummary Section

To add a variable in the ProcessDBSummary section:

**Step 1** If the required MIB attributes are not polled in the PollDefinition section, modify the polling definitions (see Modifying Poll Definitions, page 2-23).

**Step 2** If formula is required add it in the ProcessPollResult section (see Modifying the ProcessPollResult Section, page 2-24.)

**Step 3** Add the variable using *Var* as given in the following example:

**Example 1**: For variables for which formula calculation is not required, we add code in the ProcessDBSummary section as follows.

```
<ProcessDBSummary name="CPU" baseTableName="CPU">
<Var name="CPUSlot"     type="Integer" key="true">cpuSlot</Var>
</ProcessDBSummary>
```

**Example 2**: For variables for which formula calculation is required, add formulas in the ProcessPollResult section (See Modifying the ProcessPollResult Section, page 2-24) and to the ProcessDBSummary section as shown below:

```
<ProcessPollResult>

  fiveMinUtil       = cpmCPUTotal5minRev / 100;

</ProcessPollResult>

<ProcessDBSummary name="CPU" baseTableName="CPU">

       <Var name="CPUUtilMax5min" type="Double" operation="Max">fiveMinUtil</Var>
       <Var name="CPUUtilAvg5min" type="Double" operation="Avg">fiveMinUtil</Var>
</ProcessDBSummary>
```

# Modifying Poll Definitions

To modify poll definitions:

**Step 1** Include the Criteria that defined in SystemCapability for this report.

**For example:**

```
<Criteria>CISCO_PROCESS_MIB</MIBLevel>
```

Here, CISCO_PROCESS_MIB is the capability defined in SystemCapability.xml .

**Step 2** Include the poll function in the PollDefinition section as shown below:

```
cpmCPUTotalTable = poll("cpmCPUTotalIndex,cpmCPUTotal5minRev");
```

Here cpmCPUTotalIndex, and cpmCPUTotal5minRev are the MIB attributes from CISCO_PROCESS_MIB.

# Modifying the ProcessPollResult Section

To modify the ProcessPollResult section:

**Step 1** If the required MIB attributes are not polled in the PollDefinition section, see Modifying Poll Definitions, page 2-23.

**Step 2** Write the formula and assign it to a variable.

For example, in cpu.xml, cpmCPUTotal5minRev is a polled variable and the formula to be calculated is cpmCPUTotal5minRev / 100.

**Step 3** Include the formula in the <ProcessPollResult> section as follows:

```
<ProcessPollResult>

     fiveMinUtil       = cpmCPUTotal5minRev / 100;

</ProcessPollResult>
```

fiveMinUtil is a meaningful name given by the user, which can be used in the other sections of the XML such as the ProcessDBSummary section.

# Setting Up Cross Launch of Reports in Cisco Prime Network

If you are using Cisco Prime Performance Manager with Cisco Prime Network, you can set up cross-launch of Prime Performance Manager reports from Cisco Prime Network Vision device shortcut menus.

There are two ways to do this:

- Using the Prime Performance Manager GUI
- Using the **ppm crosslaunch** CLI command

## Using the Prime Performance Manager GUI

To set up cross-launch using the Prime Performance Manager GUI:

**Step 1** From the System menu, choose **Prime Network Integration**.

**Step 2** Click the Prime Network tab.

The Prime Network Gateway page appears.

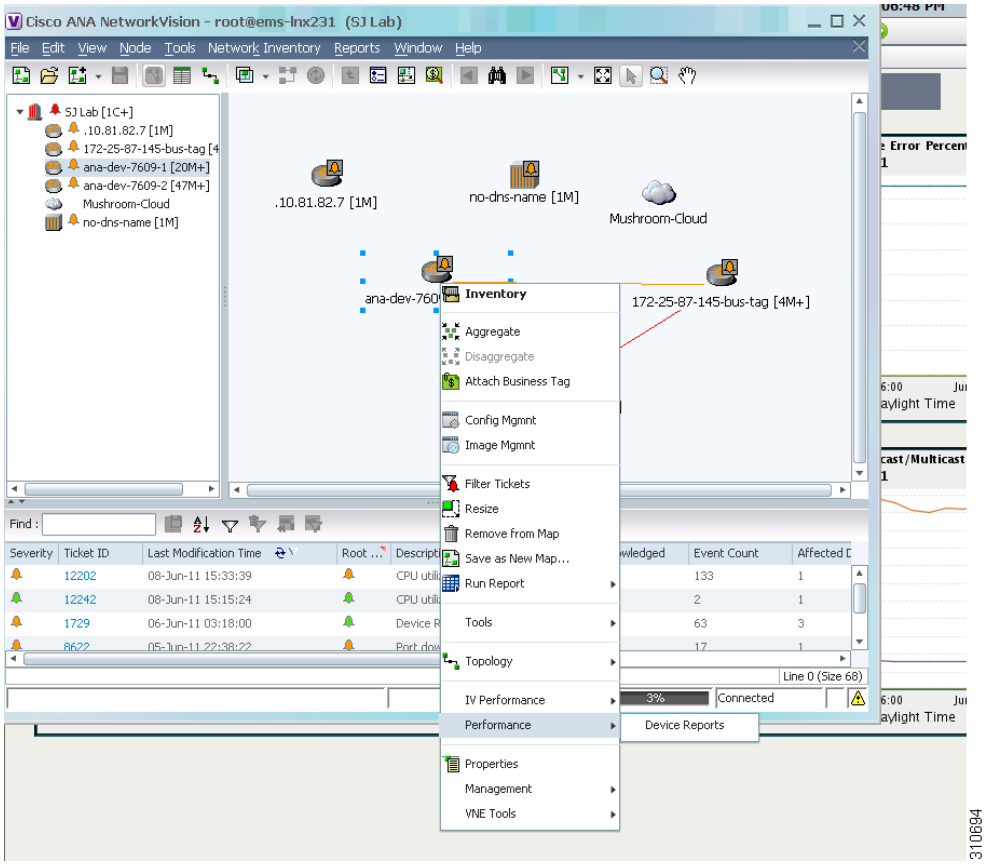
**Step 3** On the Prime Network Gateway page, enter the parameters to log into the Prime Network Gateway:

**Step 4** Click the Install Cross Launch icon.

The Prime Performance Manager gateway communicates with the Prime Network gateway and automatically installs BQL scripts that add menu selections for Cisco Prime Performance Manager reports to the Device Shortcut Menu.

Figure 2-2 shows an example:

***Figure 2-2 Device Shortcut Menu with Cross-Launch Menu Selections***




The process adds a menu select for **Performance > Device Reports** to the Prime Network device shortcut menu.

Prime Network users can now launch reports from the node level or interface level from nodes or interfaces that support the installed reports.

## Using the Command Line

To set up cross-launch using the Prime Performance Manager CLI, enter the following command on the gateway:

```
ppm crosslaunch
```

# Adding Cross Launch for Your Own Reports

The *CSCOpm-gw/etc/bql/xl* directory on the Prime Performance Manage gateway contains several BQL examples for setting and deleting cross-launch points in Prime Network.

***Table 2-1 BQL Cross Launch Scripts Provided with Cisco Prime Performance Manager***

| Script Names | Description |
|---|---|
| l2vpn-atm.bql<br>l2vpn-atm-remove.bql | Add, remove cross-launch point for L2VPN ATM Reports |
| l2vpn-epl-evpl.bql<br>l2vpn-epl-evpl-remove.bql | Add, remove cross-launch point for L2VPN EPL EVPL Reports |
| l2vpn-frame-relay.bql<br>l2vpn-frame-relay-remove.bql | Add, remove cross-launch point for L2VPN Frame Relay Reports |
| l2vpn-tdm.bql<br>l2vpn-tdm-remove.bql | Add, remove cross-launch point for L2VPN TDM Reports |
| l2vpn-vpls.bql<br>l2vpn-vpls-remove.bql | Add, remove cross-launch point for L2VPN VPLS Reports |
| l3vpn-logical.bql<br>l3vpn-logical-remove.bql | Add, remove cross-launch point for L3 VPN VRF Reports Reports |
| l3vpn.bql<br>l3vpn-remove.bql | Add, remove cross-launch point for L3 VPN Reports |
| mpls-in-segment.bql<br>mpls-in-segment-remove.bql | Add, remove cross-launch point for MPLS In Segment Reports |
| mpls-interface.bql<br>mpls-interface-remove.bql | Add, remove cross-launch point for MPLS Interface Reports |
| mpls-ldp.bql<br>mpls-ldp-remove.bql | Add, remove cross-launch point for MPLS LDP Reports |
| mpls-out-segment.bql<br>mpls-out-segment-remove.bql | Add, remove cross-launch point for MPLS Out Segment Reports |
| chassis.bql<br>chassis-remove.bql | |
| hostServer.bql<br>hostServer-remove.bql | |
| module.bql<br>module-remove.bql | |
| virtual-machine.bql<br>virtual-machine-remove.bql | |

Every BQL script that sets a cross-launch point must have a corresponding BQL file to delete the cross launch. The following examples show cross-launch BQL to add a cross-launch point at the node level in Prime Network.

Example 2-10 shows how to add a cross-launch point giving users a **Performance > Device Reports** menu selection for launching Prime Performance Manager reports at the node level.

***Example 2-10 The node.bql File***

```
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>
                <ID
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedElement)(Name=deviceRepor
t)]}</ID>
                <MenuCaption type="String">Device Reports</MenuCaption>
                <MenuPath type="String">Performance</MenuPath>
                <LineToExecute
type="String">$ppmProtocol$://$ppmWebAddress$:$ppmWebPort$/ppm/jsp/navMain.jsp?displayType
=reportTab&amp;FQDN=Node=$com.sheer.imo.IManagedElement.DeviceName$</LineToExecute>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
        <value>true</value>
    </param>
</command>
```

Example 2-11 shows sample BQL for removing the cross-launch point.

***Example 2-11 The node-remove.bql File***

```
<command name="Delete">
    <param name="oid">
        <value>
            <management.IExternalLaunch>
                <ID
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedElement)(Name=deviceRepor
t)]}</ID>
            </management.IExternalLaunch>
        </value>
    </param>
</command>
```

After you develop your own reports, you can add cross-launch points in Prime Network by modifying the provided BQL samples and then enabling cross launch.


# Creating Group Reports

The following topics tell you how to create group reports:



## Group Report Overview

A group is a construct used in Prime Performance Manager for filtering objects. The filter can be a static list of objects or it can be an algorithm that acts on a list of attributes contained in the current context. Group syntax is shown below:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<ns2:GroupList xmlns:ns2="http://cisco.com/ppm/poller">
<Group Id="1469001" editable="true" enabled="true" name="andyByName">
<Processing>
            Algorithm = If(Contains(DeviceName(), "ppm-cls"), true, false);
            Objects = ProcessorList( [ "Node=ems2941j",
                "Node=ems7606e",
                "Node=ems7606c",
                "Node=ems7609d" ] );
            Type = "Interface";
            Usage = ProcessorList( [ "AGG_INTERFACE",
                "GROUP_INTERFACE" ] );
</Processing>
</Group>
</ns2:GroupList>
```

The group report syntax includes the following elements:

- Id—The Id is assigned by Prime Performance Manager at group creation. It is essentially sgmid and uniquely identifies the group.
- Editable— (true or false) When false the group cannot be altered. This is intended for system level groups provided in the installation package for use by high level Prime Performance Manager functions. If set to true, a user with the appropriate permissions can edit the group to alter the filtering behavior.

- Enabled— (true or false) If set to false, the group does not return data to callers. If set to true, the group processes the input data and returns the appropriate values.
- Name—is assigned by the group creator and is the name displayed in the GUI.
- Algorithm—Defines how the group uses variables in the current context to determine if the current row should be included in the group processing. The algorithm uses the Prime Performance Manager reporting syntax; the same operators and macros used to define reports are also available for use in the algorithm. The final output of the algorithm should be a value of true or false.
- Objects—Objects are a static list of fully qualified domain names (FQDNs_ that map to objects in Prime Performance Manager. FQDNs can be specified the same way they are specified for the REST API interface, for example:

  ```
  Node=ems7609e
  Node=ems7609e,ifDescr=CEM1/0/7
  Node=ems7609e
  Node=ems7604h,CPUSlot=1,CPUNum=1,CPUDescr=CPU of Routing Processor
  ```

  Objects contained in this list will cause the group to pass a positive response so the current row is included in the group processing .

  The Objects and the Algorithm constructs have an OR relationship where if the current matches the algorithm OR the objects the group, a positive response is generated.
- Type—Adds a type tag to the data that passes the group. For example, a group that filters data at the network level might specify a type of Network, while a group that filters data to a particular region might specify a type of Regional. Differentiating between the two is useful when reviewing report data.
- Usage - This construct indicates what DBSummaryProcessor or AggregatedDBSummary should use this group when processing statistics. This is not a required parameter if the group is intended to be used solely by threshold processing.
- DBSummaryProcessor - this processor is used to aggregate data within the context of a single device. For instance this could aggregate all the traffic for the gigabitethernet interfaces on a single device.
- AggregatedDBSummary - this processor is used to aggregate data across multiple devices. For instance this could aggregate all the traffic for the gigabitethernet interfaces in a region for multiple devices.

# Context

The context is defined by the location calling a group. A group can be called to run from one of the following:

- AggregatedDBSummary—When a group is used in an AggregatedDBSummary context, the list of attributes available to the algorithm consist of the attributes defined in the DBSummaryProcessor to which the AggregatedDBSummary is listening.
- DBSummaryProcessor—When a group is used in the context of a DBSummaryProcessor, the list of attributes consist of the attributes defined in the ProcessPollResult to which the DBSummaryProcessor is listening.
- ThresholdProcessor—When a group is used in the context of a ThresholdProcessor, the list of attributes available to the algorithm consist of the attributes defined in the DBSummaryProcessor or AggregatedDBSummary that is the root of the threshold.

# Group Reports

A group report aggregates statistics from multiple objects into a single row identified by the group name and group type. Group report types are InterDevice and IntraDevice.

### Intradevice Reports

Intradevice reports process only objects within a single device. These can be identified by the DBSummaryProcessor definition. The DBSummaryProcessor has the following columns defined:

```
<Var name="Group" type="Group" />
<Var name="GroupType" type="GroupType" />
```

For intradevice aggregations, aggregation data is available within the scope of a single DBSummaryProcessor section. The DBSummaryProcessor section is executed within the scope of a single device. The data is available to be written to the database and can written in a timely manner like other report data. Intradevice report data is stored on the unit processing the associated device.

### Interdevice Reports

InterDevice reports process objects across the scope of multiple devices and are identified by the AggregatedDBSummary definition. Because interdevice reports can span devices, all report processing and data storage occurs on the gateway. After the unit DBSummaryProcessors finalize their data, the data is forwarded to the gateway. The gateway AggregatedDBSummary applies any groups defined for it to the data.

The data for interdevice aggregation is not available within a single DBSummaryProcessor section, so the data is processed by an AggregatedDBSummary section. This section listens to DBSummaryProcessor output. Because data is aggregated for multiple devices, AggregatedDBSummary does not know when the complete data set is available. Devices contributing to an AggregatedDBSummary can be polled at different times. Therefore, Prime Performance Manager waits for a specified time before assuming all devices are polled. This can delay data generation. You can adjust the amount time to wait before finalizing and writing results using the AGG_MAX_DELAY system property.  The default is:

```
aggMaxDelay = Long.getLong("AGG_MAX_DELAY", 3600000L);
```

Setting this lower can cause aggregations to be incomplete because some devices might not be polled within the specified interval, so they would be excluded from the aggregation.

For an AggregatedDBSummary report, a Count variable is added to the generated data row. This variable counts the number of objects contributing to the aggregated data. Knowing the object count is useful when comparing the aggregated data across multiple reporting intervals. For example, a group might aggregate all the FastEthernet interfaces in a network. The first reporting interval might have 15 devices with 30 total FastEthernet interfaces. The count would be 30 for the first interval. During the second interval a new device might be discovered that has two more FastEthernet interfaces. Subsequent intervals would have a count of 32. If two devices become unreachable for a  period of time greater than the AGG_MAX_DELAY and each of these devices have two Fast Ethernet interfaces, the intervals for the period when the devices are unreachable would have a count of 28.

**Note** The Count variable is not available for DBSummaryProcessor aggregated reports because the counts can be determined by inspecting other related reports.

# Intradevice Report Examples

The following example extracted from the interface.xml file shows some of the features and best practices. Here is the normal ProcessDBSummary definition for a non-aggregated report:

```
<ProcessDBSummarybaseTableName="INTERFACE" dbnum="0">

<Var name="ifDescr" type="String" key="true">ifFormattedName</Var>
<Var type="Integer">ifIndex</Var>
<Var type="Integer">ifType</Var>
<Var type="Double">txSpeed</Var>
<Var type="Long"   operation="Sum">txBytes</Var>
<Var type="Double" operation="Avg">txUtil</Var>
…
…  This portion removed in the interest of space
…
</ProcessDBSummary>
```

Here is an intradevice aggregation ProcessDBSummary definition:

```
<ProcessDBSummary name="GROUP_INTERFACE" baseTableName="GROUP_INTERFACE">

<Var name="Group" type="Group" />
<Var name="GroupType" type="GroupType" />

<Var type="Long"   operation="Sum">txBytes</Var>
<Var type="Double" operation="Avg">txUtil</Var>
…
… This portion removed in the interest of space
…
</ProcessDBSummary>
```

In the above example, the following were removed from the interface definition:

```
<Var name="ifDescr" type="String" key="true">ifFormattedName</Var>
<Var type="Integer">ifIndex</Var>
<Var type="Integer">ifType</Var>
<Var type="Double">txSpeed</Var>
```

These variables are not relevant for aggregations because multiple ifIndex, ifDescr, ifType, and txSpeed values could be involved. In the above example, the following columns are defined:

```
<Var name="Group" type="Group" />
<Var name="GroupType" type="GroupType" />
```

These columns define the report as an intradevice aggregation. When the DBSummaryProcessor sees these fields it knows to process it as an aggregation and also to send the group name and group type as key fields .

The CSV definition, shown below, is the next XML definition related to the intradevice aggregation.

```
<CSV name="GROUPINTERFACE" location="gateway" listen="GROUP_INTERFACE" aggregate="true">

<Column name="GroupType">GroupType</Column>

<Column name="SendTotalPkts">txTotPkts</Column>
<Column name="SendTotalPktRate">txTotPkts / IntervalDuration()</Column>
…
… This portion removed in the interest of space
…
</CSV>
```

In the example, the CSV processor listens to the defined GROUP_INTERFACE processor. Because aggregate=true, the CSV processor generates a CSV file based on the aggregated results. No reference to the group name is provided because in aggregated CSV report files the group name is a key field and therefore it is automatically included as part of the data.

The WebReport definitions are the next xml definitions in intradevice aggregations. In the following example, items of interest include:

- category—In this example, IntraDevice is broken into its own level in the tree because of the large number of reports. For smaller report trees, placing intradevice and interdevice group reports at the same tree level is acceptable.
- reportId—Be careful when using this parameter because it can cause problems with report activation. To control group reports separately from normal reports and to identify interdevice reports from intradevice reports, each report type should have a unique reportId. In thePrime Performance Manager interface three reportids are specified in the Poll construct in the interface.xml file to specify the appropriate entry for the individual webreports:

```
<Poll name="Interface"  reportId="INTERFACE,INTERFACEINTRA,INTERFACEINTER">
```

- context—Allows the report to be placed under the Grouped Reports branch of the navigation tree. “Group” is the indicator that determines the report position in the Grouped Reports tree.
- baseTable="GROUP_INTERFACE"—Similar to other reports, the DB table the webreport is based on must be specified.
- Link—Specifies the hyperlinks to display in the webreport for the summary table, tableview, and the legend.

The following shows an example intradevice report:

```
<Link name="node"      context="Node">fqdnid</Link>
<Key name="group"   context="Group">Group</Key>
<Key name="groupType"   context="groupType">GroupType</Key>

<WebReport name="wrnIfBroadcastPercentGroup"
category="level1Transport,level2Interface,level3InterfaceIntraDevice"
reportId="INTERFACEINTRA"
context="Network,Group"
textProps="interface"
sortWeight="16">

<Criteria>IF_MIB</Criteria>

<GraphView>
<GraphSummary title="ifBroadcastPercentageGroup" />
<Graph title="ifAvgTotalBroadcastPercentage">
<Util name="broadcastPercentage">(txBcastPktPercent + rxBcastPktPercent) / 2</Util>
</Graph>
<Graph title="ifAvgSendBroadcastPercentage">
<Util name="sendBroadcastPercentage">txBcastPktPercent</Util>
</Graph>
<Graph title="ifAvgRecvBroadcastPercentage">
<Util name="recvBroadcastPercentage">rxBcastPktPercent</Util>
</Graph>

<LeafGraph title="ifAvgSendRecvBroadcastPercentageGroup">
<Util name="ifAvgTotalBroadcastPercentage">(txBcastPktPercent + rxBcastPktPercent) /
2</Util>
<Util name="ifAvgSendBroadcastPercentage">txBcastPktPercent</Util>
<Util name="ifAvgRecvBroadcastPercentage">rxBcastPktPercent</Util>
</LeafGraph>
</GraphView>
```

```
<TableViewbaseTable="GROUP_INTERFACE">
<IdLabel/>
<Label colSpan="1" name=""/>
<Label colSpan="2" name="sendPackets"/>
<Label colSpan="3" name="sendPercentage"/>
<Label colSpan="2" name="recvPackets"/>
<Label colSpan="3" name="recvPercentage"/>
<HeaderRow/>
<Link name="node"      context="Node">fqdnid</Link>
<Key name="group"   context="Group">Group</Key>
<Key name="groupType"   context="groupType">GroupType</Key>

<Time/>

<Column name="allcast">txTotPkts</Column>
<Column name="allcastSec">txTotPkts / IntervalDuration()</Column>
<Util   name="unicast">txUcastPktPercent</Util>
<Util   name="multicast">txMcastPktPercent</Util>
<Util   default="true" name="broadcast">txBcastPktPercent</Util>
<Column name="allcast">rxTotPkts</Column>
<Column name="allcastSec">rxTotPkts / IntervalDuration()</Column>
<Util   name="unicast">rxUcastPktPercent</Util>
<Util   name="multicast">rxMcastPktPercent</Util>
<Util   name="broadcast">rxBcastPktPercent</Util>

</TableView>
</WebReport>

?
```

# Interdevice Reports

The following example from the interface.xml file shows interdevice features and best practices. The following section is the ProcessDBSummary definition for a non-aggregated report that is the source of data for an AggregatedDBSummary section.

```
<ProcessDBSummarybaseTableName="INTERFACE" dbnum="0">

<Var name="ifDescr" type="String" key="true">ifFormattedName</Var>
<Var type="Integer">ifIndex</Var>
<Var type="Integer">ifType</Var>
<Var type="Double">txSpeed</Var>
<Var type="Long"   operation="Sum">txBytes</Var>
<Var type="Double" operation="Avg">txUtil</Var>
…
… This portion removed in the interest of space
…
</ProcessDBSummary>
```

The following example is the AggregatedDBSummary section for an interdevice aggregation definition. The AggregatedDBSummary listens for data from the INTERFACE ProcessDBSummary as input.

```
<AggregatedDBSummarybaseTableName="AGG_INTERFACE"
    location="gateway" maxEntriesPerId="100"
listen="INTERFACE">

<Var type="Long"   operation="Sum">txBytes</Var>
<Var type="Double" operation="Avg">txUtil</Var>
…
… This portion removed in the interest of space
```

```
…
</AggregatedDBSummary>
```

In the above example, the following items were removed from the INTERFACE definition:

```
<Var name="ifDescr" type="String" key="true">ifFormattedName</Var>
<Var type="Integer">ifIndex</Var>
<Var type="Integer">ifType</Var>
<Var type="Double">txSpeed</Var>
```

These variables are not relevant for aggregations because multiple ifIndex, ifDescr, ifType, and txSpeed values could be involved.The AggregatedDBSummary code knows to send group name and group type as key fields. In addition, a count field is included. This field contains the count of objects that contributed to the aggregation. If seven interfaces passed the group filtering, the count variable is set to 7.

The CSV definition is the next interdevice aggregation xml definition.

```
<CSV name="AGGINTERFACE" location="gateway" listen="AGG_INTERFACE" aggregate="true">

<Column name="GroupType">groupType</Column>
<Column name="Count">count</Column>

<Column name="SendTotalPkts">txTotPkts</Column>
<Column name="SendTotalPktRate">txTotPkts / IntervalDuration()</Column>
…
… This portion removed in the interest of space
…
</CSV>
```

In the above example, the CSV processor listens to the defined AGG_INTERFACE processor. The aggregate="true" attribute causes the CSV processor to generate a CSV file based on the aggregated results.

No reference to the group name is provided because in aggregated report CSV files group name is the key field and automatically included in the data.

WebReport definitions are the next intradevice aggregation xml definitions. Items of interest include:

- category—In this example, InterDevice is placed in its own tree level because the number of the large number of reports. For smaller report trees, placing intradevice and interdevice group reports at the same navigation tree level is acceptable.
- reportId—Use this this parameter with care because it can cause report activation problems. To control group reports separately from normal reports and interdevice reports from intradevice reports, each report type should have a unique reportId. In the interface example, three reportids on the Poll construct are added in the interface.xml file to specify the appropriate entry for the individual web reports:

```
<Poll name="Interface"  reportId="INTERFACE,INTERFACEINTRA,INTERFACEINTER">
```

- context—Allows the report to be placed under the Grouped Reports section of the Prime Performance Manager navigation tree. “Group” is the indicator that sets this.
- baseTable="AGG_Interface"—All reports must specify the DB table on which the webreport is based.
- Link—Specifies the hyperlinks to display in the webreport for the summary table, tableview, and the legend

  ```
  <Linkname="group"   context="Group">Group</Link>
  <Key name="groupType"   context="groupType">GroupType</Key>
  ```

- Count—Is an additional field to add to the AggregatedDBSummary-based web reports

An example interdevice group report is shown below:

```
<Column name="Count" thresholdable="false">count</Column>

<WebReport name="wrnIfBroadcastPercentAgg"
category="level1Transport,level2Interface,level3InterfaceInterDevice"
reportId="INTERFACEINTER"
context="Network,Group"
textProps="interface"
sortWeight="16">

<Criteria>IF_MIB</Criteria>

<GraphView>
<GraphSummary title="ifBroadcastPercentageAgg" />
<Graph title="ifAvgTotalBroadcastPercentage">
<Util name="broadcastPercentage">(txBcastPktPercent + rxBcastPktPercent) / 2</Util>
</Graph>
<Graph title="ifAvgSendBroadcastPercentage">
<Util name="sendBroadcastPercentage">txBcastPktPercent</Util>
</Graph>
<Graph title="ifAvgRecvBroadcastPercentage">
<Util name="recvBroadcastPercentage">rxBcastPktPercent</Util>
</Graph>

<LeafGraph title="ifAvgSendRecvBroadcastPercentageAgg">
<Util name="ifAvgTotalBroadcastPercentage">(txBcastPktPercent + rxBcastPktPercent) /
2</Util>
<Util name="ifAvgSendBroadcastPercentage">txBcastPktPercent</Util>
<Util name="ifAvgRecvBroadcastPercentage">rxBcastPktPercent</Util>
</LeafGraph>
</GraphView>

<TableViewbaseTable="AGG_Interface">
<IdLabel/>
<Label colSpan="1" name=""/>
<Label colSpan="2" name="sendPackets"/>
<Label colSpan="3" name="sendPercentage"/>
<Label colSpan="2" name="recvPackets"/>
<Label colSpan="3" name="recvPercentage"/>
<HeaderRow/>
<Link name="group"       context="Group">fqdnid</Link>
<Key name="groupType"   context="groupType">groupType</Key>
<Time/>

<Column name="Count" thresholdable="false">count</Column>

<Column name="allcast">txTotPkts</Column>

<Column name="allcastSec">txTotPkts / IntervalDuration()</Column>

<Util   name="unicast">txUcastPktPercent</Util>
<Util   name="multicast">txMcastPktPercent</Util>

<Util   default="true" name="broadcast">txBcastPktPercent</Util>

<Column name="allcast">rxTotPkts</Column>

<Column name="allcastSec">rxTotPkts / IntervalDuration()</Column>

<Util   name="unicast">rxUcastPktPercent</Util>
<Util   name="multicast">rxMcastPktPercent</Util>
<Util   name="broadcast">rxBcastPktPercent</Util>
```

```
</TableView>

</WebReport>
```

# Group by Group

The Prime Performance Manager GroupBy groups construct is primarily intended for small cell network support. To set up GroupBy groups:

The $GW/etc/apDrop/ directory contains a file called, “usage”. It contains all the base table names for which Prime Performance Manager generates groups. You can configure this list as needed. By default, it is populated with all the RAN Management System (RMS) Access Point (AP) aggregate base table names.

In the $GW/etc/apDrop/ directory, create a MYGROUP.groupby file. MYGROUP will be the name of the generated group, the .groupby suffix tells the parser how to read it.

In MYGROUP.groupby, you can add a new line for each Group By that you want to create. The first part of the line is the name of the Group By, followed by a ',' and then the algorithm to use to derive the group name. So, :

```
Hardware Version,HWVersion
Area and Site,Area + " " + Site
Software Version,${SW Version}
```

Everything on the right of the first comma is an algorithm that works on the variables from ProcessDBSummary and from the AP Inventory (so, any variable Prime Performance Manager is collecting the GDDT tool). If the variable has a space in it (some of the variables from the GDDT tool do), you can wrap it in a ${} so it treats everything inside the {} as the variable name, even if it has spaces. <<Note: The AP Inventory and GDDT tool data are all specific to the Cisco Small Cell solution>>

Save this file, then run:

```
$GW/bin/ppmGroupGenerator.sh MYGROUP.groupby
```

The script should print out a success message. In the $GW/etc/groups/user/ directory, you should see a MYGROUP.xml file. As new data is collected, Prime Performance Manager automatically begins generating groups based on these definitions.


# Testing and Debugging Your Report

The following topics provide procedures that you can use to test your report and resolve common issues and errors:



## Testing Your Report

To test your report:

**Step 1** Save your report and its associated .properties file in the /opt/CSCOppm-gw/etc/pollers/user directory.

**Step 2** In the Prime Performance Manager GUI, select **Reports** and then click the Reports Status tab.

**Step 3** Scroll down the list of reports and locate your report.

**Step 4** Check the interval(s) for which you want to enable your report then click the **Save** icon.

The report should now be active.

**Step 5** Navigate to the report category associated with your report, then click your report.

**Step 6** Note any error messages that appear.

**Step 7** If you have set up cross-launch of Cisco Prime Performance Management reports:

**a.** Log into Cisco Prime Network Vision.

**b.** Bring up a network map.

**c.** Locate the device or network location where your cross-launch is set up.

**d.** Right click on the cross-launch point.

**e.** Verify that the NetworkVision Device Shortcut has a Performance select; for example, **Performance > Device Reports**.

**f.** Click **Device Reports** and verify that a Cisco Prime Performance Manager report comes up.

**g.** Verify that any reports you set up to cross-launch from Prime Network are available and work correctly.

# Common Issues and Error Messages

This section lists common issues and error messages that can occur when running reports and provides information on how to resolve the problems.

To see error messages:

**Step 1** From the System menu, choose **Messages**.

**Step 2** To view only error messages, click **Error**.

**Step 3** To see a console log, under the System menu, choose **Logs**, then choose **Console Log**.

## Incorrect MIB Variable

If your report uses an incorrect MIB variable, an error message will appear. The incorrect MIB variable will appear in the sgmConsoleLog.txt or messageLog.txt file. These log files can also be viewed from the System->Logs and System->Messages menus.

Check the MIB and check the XML code in your report to make sure that the variable is referenced correctly.


# Prime Performance Manager Event API

Prime Performance Manager provides Remote Procedure Call (RPC) Event API that accepts Simple Object Access Protocol (SOAP) OSS requests and responds with SOAP responses and traps. Prime Performance Manager also sends unsolicited messages through traps to the OSS. You can configure Prime Performance Manager to send new events to the northbound OSS asynchronously through traps.

The following topics describe the Prime Performance Manager events API:



## Overview to Events and Alarms

An event is a single occurrence at a specific moment in time. Each event has an event ID. An alarm is a sequence of events that occur over a period of time. An alarm has a single alarm ID that exists for the duration of the events that define the alarm. For example, when the chassis temperature exceeds a certain threshold, Prime Performance Manager reports a minor alarm. When the temperature increases again, Prime Performance Manager escalates the alarm to major. When the temperature increases a third time, Prime Performance Manager escalates the alarm to critical. However, the alarm ID for this event sequence remains the same.

Prime Performance Manager does not consider the clearing condition in a state transition sequence like the one described in the example. For example, an ITP link may change state from normal to critical to normal to warning within an hour. An event is created for each transition: from normal to critical, from critical to normal, and from normal to warning. these three events make up the event sequence of an ITP Link State alarm.

The system or a user deletes (archives) alarms. Cleared alarms are alarms that have the severity Normal. The system archives clears alarms after one day (this is the default setting). The system archives unclears alarms after seven days. After the system or a user archives an alarm, if the condition occurs again, Prime Performance Manager raises a new alarm with a new alarm ID.

### Configuring OSS Hosts

Configuring Prime Performance Manager to send events to a northbound OSS is performed using the Prime Performance Manager GUI. For information see "Adding Upstream OSS Hosts" procedure in the *Cisco Prime Performance Manager 1.7 User Guide*.

# Prime Performance Manager Event API Operations

The following topics describe Prime Performance Manager event API operations. All the operations are listed as pseudocode with comments. The operations syntax is defined as Web Services Description Language (WSDL). The syntax is described in Event API WSDL and XSD Definitions, page 5-8. Event API operations use SOAP (Simple Object Access Protocol). Error codes are described in Event API Errors, page 5-16.

Event API operations:



# Get all Open Alarms from Prime Performance Manager

```
int getAllOpenAlarmsAsTraps (TrapTarget target)
```

This method retrieves all the open alarms from Prime Performance Manager as traps.

### Parameters

*TrapTarget target*—Specifies the target to send the Prime Performance Manager alarm traps. The following parameters can be specified:

*Hostname*—Hostname or IP address to send the traps to.

*Port Number* —Port number to send the traps to.

*Community String*—Community string to fit the trap.

*SNMP Version*—Simple Network Management Protocol (SNMP) version for the traps: 1 or 2c.

*MIB*—Management Information Base (MIB) format to send the traps: CISCO-PRIME, CISCO-SYSLOG or CISCO-EPM-2.

#### Return Value

Number of open alarms sent as a result of this method.

## Get all Events from Prime Performance Manager

```
int getAllEventsAsTraps (TrapTarget target)
```

This method retrieves all the events from Prime Performance Manager as traps.

#### Parameters

*TrapTarget target*—Specifies the target to send the Prime Performance Manager event traps. The following parameters can be specified:

*Hostname*—Hostname or IP address to send the traps to.

*Port Number* —Port number to send the traps to.

*Community String*—Community string to fit the trap.

*SNMP Version*—Simple Network Management Protocol (SNMP) version for the traps: 1 or 2c.

*MIB*—Management Information Base (MIB) format to send the traps: CISCO-PRIME, CISCO-SYSLOG or CISCO-EPM-2.

#### Return Value

Number of events sent as a result of this method.

## Get Filtered Events from Prime Performance Manager

```
EventList getFilteredEvents (EventFilter filter)
```

This method retrieves a list of filtered events from Prime Performance Manager.

#### Parameters

- EventFilter—Specifies the rules to retrieve Prime Performance Manager events. Filters can be standalone or combined. If multiple filters are specified, they are applied using AND logic. The following parameters can be specified:
  - Event ID—Specifies a list of event IDs to filter.
  - Start Date—Specifies the starting date to filter the events.
  - End Date—Specifies the end date to filter the events.
  - Severity—Specifies a list of severities to filter the events. Valid severities can be customized in the Prime Performance Manager Event Editor.
  - Category—Specifies a list of categories to filter the events. Valid event categories can be customized in the Prime Performance Manager Event Editor.
  - Acknowledged—Filter based on whether the events are acknowledged.
  - Cleared—Filter based on whether the events are cleared.
  - Message Text—Filter based on whether the events contain a given message text.
  - StartAlarmId—Filter based on the starting alarm ID.
  - EndAlarmId—Filter based on the ending alarm ID.

  - StartAlarmChangeTime—Filter based on the starting alarm change time.
  - EndAlarmChangeTime—Filter based on the ending alarm ID.
  - NetworkElement—Filter based on the network element.
  - Tenant—Filter based on the tenant.
- Forward—Filter based on whether the forward option is turned on for an event. Forward option for the events is configured using the Prime Performance Manager Event Editor.
- AlarmMode—Alarms and events filter:
  - 2—Filters for alarms, for example, <AlarmMode>2</AlarmMode>
  - 3—Filters for events, for example <AlarmMode>3</AlarmMode>

#### Return Value

A list of events sent as a result of this method.

#### Return Value

```
<message name="getFilteredEventsRequest">
    <part name="filter" type="tns:EventFilter"></part>
  </message>
  <message name="getFilteredEventsResponse">
    <part name="events" type="tns:EventList"></part>
  </message>
```

# Get Filtered Events as Traps from Prime Performance Manager

**`int getFilteredEventsAsTraps (TrapTarget target, EventFilter filter)`**

This method retrieves the filtered events from Prime Performance Manager as traps.

#### Parameters

*TrapTarget target*—Specifies the target to send Prime Performance Manager event traps. The following parameters can be specified:

- *Hostname*—Hostname or IP address to send the traps to.
- *Port Number*—Port number to send the traps to.
- *Community String*—Community string to fit the trap.
- *SNMP Version*—SNMP version for the traps: 1 or 2c.
- *MIB*—MIB format to send the traps: CISCO-PRIME. CISCO-SYSLOG or CISCO-EPM-2.

*EventFilter filter*—Specifies the filter rules to retrieve Prime Performance Manager event. These filters can be specified as standalone or combined together. If multiple filters are specified, they are applied using “AND” logic. The following parameters can be specified:

- *Event ID*—Specifies a list of event IDs to filter.
- *Start Date*—Specifies the starting date to filter the events.
- *End Date*—Specifies the end date to filter the events.
- *Severity*—Specifies a list of severities to filter the events. Valid severities can be customized in the the Prime Performance Manager Event Editor.

- *Category*—Specifies a list of categories to filter the events. Valid event categories can be customized in the Prime Performance Manager Event Editor.
- *Acknowledged*—Filter based on whether the events are acknowledged.
- *Cleared*—Filter based on whether the events are cleared.
- *Message Text*—Filter based on whether the events contain a given message text.
- *Forward*—Filter based on whether the forward option is turned on for an event. Forward option for the events is configured using the Prime Performance Manager Event Editor.
- *AlarmMode*—Filter based on alarms or events.
- StartAlarmId—Filter based on the starting alarm ID.
- EndAlarmId—Filter based on the ending alarm ID.
- StartAlarmChangeTime—Filter based on the starting alarm change time.
- EndAlarmChangeTime—Filter based on the ending alarm ID.
- NetworkElement—Filter based on the network element.
- Tenant—Filter based on the tenant.

#### Return Value

Number of events sent as a result of this method.

## Resend Traps

```
void resendTraps(String host, String port)
```

This method resends the traps.

#### Parameters

- host—Trap destination host.
- port—Trap destination port.

#### Returns

void

#### Example

```
<message name="resendTrapsRequest">
  <part name="host" type="xsd:string"/>
  <part name="port" type="xsd:string"/>
</message>
<message name="resendTrapsResponse">
</message>
```

## Clear Events

```
void clearEvents (EventIDList eventList, String userid, String note)
```

This method clears the specified events.

#### Parameters

*EventIDList eventList*—List of the events to clear.

*String userid*—User ID who cleared the events.

*String note*—Note explaining the reason for clearing this event.

#### Return Value

None

## Acknowledge Events

**`void acknowledgeEvents (EventIDList eventList, String userid, String note)`**

This method acknowledges the specified events.

#### Parameters

*EventIDList eventList*—List of the events to acknowledge.

*String userid*—User ID who cleared the events.

*String note*—Note explaining the reason for acknowledging the events.

#### Return Value

None

## Unacknowledge Events

**`void unacknowledgeEvents (EventIDList eventList, String userid, String note)`**

This method unacknowledges the specified events.

#### Parameters

*EventIDList eventList*—List of the events to unacknowledge.

*String userid*—User ID who cleared the events.

*String note*—Note explaining the reason for unacknowledging the events.

#### Return Value

void

#### Example

```
<message name="unAcknowledgeEventsRequest">
    <part name="eventList" type="tns:EventIDList"/>
    <part name="userid" type="xsd:string"/>
    <part name="note" type="xsd:string"/>
  </message>
  <message name="unAcknowledgeEventsResponse">
  </message>
```

# Delete Events

```
void deleteEvents (EventIDList eventList)
```

This method deletes the specified events.

### Parameters

*EventIDList eventList*—List of the events to delete.

### Return Value

None

# Change Event Severity

```
void changeSeverities (EventIDList eventList, String severity, String userid, String note)
```

This method changes severity of specified events.

### Parameters

*EventIDList eventList*—List of the events to change severity

*String severity*—The target severity to change. Valid severities can be customized in the Prime Performance Manager Event Editor.

*String userid*—User ID who changed the event severity.

*String note*—Note explaining the reason for changing the severity for the events.

### Return Value

None

# Get Note for an Event

```
String getNote (long eventID)
```

This method gets an attached note for an event.

### Parameters

*Long eventID*—Event ID to retrieve the note.

### Return Value

None

# Set Note for an Event

```
String setNote (long eventID, String userid, String note)
```

This method sets an attached note for an event.

#### Parameters

*long eventID*—Event ID to set the note.

*String userid*—User ID who sets the note.

*String note*—Note text to set to.

#### Return Value

None

## Append Note to an Event

**`String appendNote (long eventID, String userid, String note)`**

This method appends a note to an event.

#### Parameters

*long eventID*—Event ID to append the note to.

*String userid*—User ID who appends the text to the event note.

*String note*—Text to append to the event note.

#### Return Value

None

# Event API WSDL and XSD Definitions

The following topics provide the Prime Performance Manager Event API WSDL and XSD definitions.



## EventAPI.wsdl

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) 2006-2013 Cisco Systems, Inc.    All rights reserved. -->

<definitions targetNamespace="http://cisco.com/ppm" name="EventAPIService"
xmlns:tns="http://cisco.com/ppm"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">

<types>
<xsd:schema>
<xsd:import namespace="http://cisco.com/ppm" schemaLocation="MWTM.xsd" />
<xsd:import namespace="http://cisco.com/ppm" schemaLocation="Event.xsd" />
<xsd:import namespace="http://cisco.com/ppm" schemaLocation="Common.xsd" />
</xsd:schema>
</types>

<message name="APIStatus">
<part name="APIStatus" element="tns:APIStatus"/>
```

```
</message>
<message name="resendTrapsRequest">
<part name="host" type="xsd:string"/>
<part name="port" type="xsd:string"/>
</message>
<message name="resendTrapsResponse">
</message>
<message name="getFilteredEventsRequest">
<part name="filter" type="tns:EventFilter"></part>
</message>
<message name="getFilteredEventsResponse">
<part name="events" type="tns:EventList"></part>
</message>
<message name="getAllOpenAlarmsAsTrapsRequest">
<part name="target" type="tns:TrapTarget"/>
</message>
<message name="getAllOpenAlarmsAsTrapsResponse">
<part name="eventCount" type="xsd:int"/>
</message>
<message name="getAllEventsAsTrapsRequest">
<part name="target" type="tns:TrapTarget"/>
</message>
<message name="getAllEventsAsTrapsResponse">
<part name="eventCount" type="xsd:int"/>
</message>
<message name="getFilteredEventsAsTrapsRequest">
<part name="target" type="tns:TrapTarget"/>
<part name="filter" type="tns:EventFilter"/>
</message>
<message name="getFilteredEventsAsTrapsResponse">
<part name="eventCount" type="xsd:int"/>
</message>
<message name="clearEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="clearEventsResponse">
</message>
<message name="acknowledgeEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="acknowledgeEventsResponse">
</message>
<message name="unAcknowledgeEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="unAcknowledgeEventsResponse">
</message>
<message name="deleteEventsRequest">
<part name="eventList" type="tns:EventIDList"/>
</message>
<message name="deleteEventsResponse">
</message>
<message name="changeSeveritiesRequest">
<part name="eventList" type="tns:EventIDList"/>
<part name="severity" type="xsd:string"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
```

```
<message name="changeSeveritiesResponse">
</message>
<message name="getNoteRequest">
<part name="eventID" type="xsd:long"/>
</message>
<message name="getNoteResponse">
<part name="note" type="xsd:string"/>
</message>
<message name="setNoteRequest">
<part name="eventID" type="xsd:long"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="setNoteResponse">
</message>
<message name="appendNoteRequest">
<part name="eventID" type="xsd:long"/>
<part name="userid" type="xsd:string"/>
<part name="note" type="xsd:string"/>
</message>
<message name="appendNoteResponse">
</message>

<portType name="EventAPI">
<operation name="resendTraps">
<input message="tns:resendTrapsRequest"/>
<output message="tns:resendTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getFilteredEvents">
<input message="tns:getFilteredEventsRequest" />
<output message="tns:getFilteredEventsResponse" />
<fault name="APIStatus" message="tns:APIStatus" />
</operation>
<operation name="getAllOpenAlarmsAsTraps">
<input message="tns:getAllOpenAlarmsAsTrapsRequest"/>
<output message="tns:getAllOpenAlarmsAsTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getAllEventsAsTraps">
<input message="tns:getAllEventsAsTrapsRequest"/>
<output message="tns:getAllEventsAsTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getFilteredEventsAsTraps">
<input message="tns:getFilteredEventsAsTrapsRequest"/>
<output message="tns:getFilteredEventsAsTrapsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="clearEvents">
<input message="tns:clearEventsRequest"/>
<output message="tns:clearEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="acknowledgeEvents">
<input message="tns:acknowledgeEventsRequest"/>
<output message="tns:acknowledgeEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="unAcknowledgeEvents">
<input message="tns:unAcknowledgeEventsRequest"/>
<output message="tns:unAcknowledgeEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
```

```
<operation name="deleteEvents">
<input message="tns:deleteEventsRequest"/>
<output message="tns:deleteEventsResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="changeSeverities">
<input message="tns:changeSeveritiesRequest"/>
<output message="tns:changeSeveritiesResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="getNote">
<input message="tns:getNoteRequest"/>
<output message="tns:getNoteResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="setNote">
<input message="tns:setNoteRequest"/>
<output message="tns:setNoteResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
<operation name="appendNote">
<input message="tns:appendNoteRequest"/>
<output message="tns:appendNoteResponse"/>
<fault name="APIStatus" message="tns:APIStatus"/>
</operation>
</portType>

<binding name="EventAPIPortBinding" type="tns:EventAPI">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
<operation name="resendTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault use="literal" name="APIStatus" />
</fault>
</operation>
<operation name="getFilteredEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault use="literal" name="APIStatus" />
</fault>
</operation>
<operation name="getAllOpenAlarmsAsTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
```

```
</operation>
<operation name="getAllEventsAsTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="getFilteredEventsAsTraps">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="clearEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="acknowledgeEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="unAcknowledgeEvents">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="deleteEvents">
<soap:operationsoapAction="" />
<input>
```

```
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="changeSeverities">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="getNote">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="setNote">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
<operation name="appendNote">
<soap:operationsoapAction="" />
<input>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</input>
<output>
<soap:body use="literal" namespace="http://cisco.com/ppm" />
</output>
<fault name="APIStatus">
<soap:fault name="APIStatus" use="literal" />
</fault>
</operation>
</binding>

<service name="EventAPIService">
<port name="EventAPIPort" binding="tns:EventAPIPortBinding">
<soap:address location="REPLACE_WITH_ACTUAL_URL"/>
</port>
</service>
```

```
</definitions>
```

# Event.xsd

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) 2006-2013 Cisco Systems, Inc.    All rights reserved. -->

<xs:schema version="1.0"
targetNamespace="http://cisco.com/ppm"
xmlns:tns="http://cisco.com/ppm"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- A list of event IDs -->
<xs:complexType name="EventIDList">
<xs:sequence>
<xs:element name="ID" type="xs:long" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:element name="EventIDList" type="tns:EventIDList"/>

<!-- A list of Events -->
<xs:complexType name="EventList">
<xs:sequence>
<xs:element name="Events" type="tns:Event" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>

<xs:element name="EventList" type="tns:EventList"/>

<!-- Event : derived from com.cisco.mwg.sgm.core.SgmEvent -->
<xs:complexType name="Event">
<xs:sequence>
<xs:element name="Id" type="xs:long" />
<xs:element name="AlarmId" type="xs:long" />
<xs:element name="Owner" type="xs:string" />
<xs:element name="Category" type="xs:string" />
<xs:element name="Severity">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="Critical" />
<xs:enumeration value="Major" />
<xs:enumeration value="Minor" />
<xs:enumeration value="Warning" />
<xs:enumeration value="Informational" />
<xs:enumeration value="Normal" />
<xs:enumeration value="Indeterminate" />
<xs:enumeration value="Normal" />
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Message" type="xs:string" />
<xs:element name="NotesExist" type="xs:boolean" />
<xs:element name="AckBy" type="xs:string" />
<xs:element name="ClearBy" type="xs:string" />
<xs:element name="CreateTimestamp" type="xs:dateTime" />
<xs:element name="AckTimestamp" type="xs:dateTime" />
<xs:element name="ClearTimestamp" type="xs:dateTime" />
<xs:element name="ChangeTimestamp" type="xs:dateTime" />
<xs:element name="AlarmNature" type="xs:string" />
```

```
<xs:element name="AlarmType" type="xs:string" />
<xs:element name="ProbableCause" type="xs:string" />
<xs:element name="DeviceType" type="xs:string" />
<xs:element name="Condition" type="xs:string" />
<xs:element name="TcaName" type="xs:string" />
<xs:element name="TcaMetric" type="xs:string" />
<xs:element name="ElementName" type="xs:string" />
<xs:element name="OriginalSeverity" type="xs:string" />
<xs:element name="Count" type="xs:long" />
<xs:element name="isAlarm" type="xs:boolean" />
</xs:sequence>
</xs:complexType>

<xs:element name="Event" type="tns:Event"/>

<!-- Trap Target specifies target host/port and SNMP parameters to send
       SNMP Trap notification to -->
<xs:complexType name="TrapTarget">
<xs:sequence>
<xs:element name="Hostname" type="xs:string"/>
<xs:element name="Port" type="xs:int"/>
<xs:element name="Community" type="xs:string"/>
<xs:element name="SNMPVersion">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="1"/>
<xs:enumeration value="2c"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="MIB">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="CISCO-PRIME"/>
<xs:enumeration value="CISCO-EPM-2"/>
<xs:enumeration value="CISCO-SYSLOG"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:element name="TrapTarget" type="tns:TrapTarget"/>

<!-- Event Filter -->
<!-- If more than one conditions are specified,
EventFilter applies "AND" on all specified conditions
   -->
<xs:complexType name="EventFilter">
<xs:sequence>
<xs:element name="EventIDs" type="tns:EventIDList" minOccurs="0"/>
<xs:element name="StartAlarmId" type="xs:int" minOccurs="0"/>
<xs:element name="EndAlarmId" type="xs:int" minOccurs="0"/>
<xs:element name="StartAlarmChangeTime" type="xs:dateTime" minOccurs="0"/>
<xs:element name="EndAlarmChangeTime" type="xs:dateTime" minOccurs="0"/>
<xs:element name="StartDate" type="xs:dateTime" minOccurs="0"/>
<xs:element name="EndDate" type="xs:dateTime" minOccurs="0"/>
<xs:element name="Severity" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="Category" type="xs:string"
minOccurs="0" maxOccurs="unbounded"/>
<xs:element name="Acknowledged" type="xs:boolean" minOccurs="0"/>
<xs:element name="Cleared" type="xs:boolean" minOccurs="0"/>
<!-- filter "text" is based on whether event message contains
```

```
given text -->
<xs:element name="MessageText" type="xs:string" minOccurs="0"/>
<xs:element name="NetworkElement" type="xs:string" minOccurs="0"/>
<xs:element name="Forward" type="xs:boolean" minOccurs="0"/>
<xs:element name="AlarmMode" type="xs:int" minOccurs="0"/>
</xs:sequence>
</xs:complexType>

<xs:element name="EventFilter" type="tns:EventFilter"/>

</xs:schema>
```

# Event API Errors

Detailed error information is defined as APIStatus in WSDL (for Event API WSDL definitions, see Appendix 5, "Event API WSDL and XSD Definitions"):

```
<xs:complexType name="APIStatus">
  <xs:sequence>
    <xs:element name="StatusCode" type="xs:int"/>
    <xs:element name="Message" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

APIStatus contains a status code and a message. Table 5-1 lists the possible Event API status codes.

***Table 5-1 Status Codes***

| Status Code | Error | Description |
|---|---|---|
| 1000 | UNEXPECTED_ERROR | An unexpected error occurred that stops the requested operation. |
| 2000 | INVALID_PARAMETER | The requestor provided one or more invalid parameters in the requested operation. |

# Cross Launching Prime Performance Manager

To cross launch into Prime Performance Manager, generate URLs in the following format:

**<server>:<port>/ppm/jsp/main.jsp?**

Required parameters:

- displayType [results,reportTab]—Shows only the table and graphs; reportTab also includes the header bar. This parameter shows the entire Prime Performance Manager inner frame without the XWT shell.
- FQDN [objectId]—Specifies an object by FQDN.
- resultsType <string>—The report to run. The string must match the list provided in the Type field in the Prime Performance Manager GUI, for example, Interface Utilization Hourly. You must specify a reportID with either FQDN or resultsType.

Optional Parameters:

- durationSelect <string>—Sets a report duration. You can use this parameter instead of a start date and end date to the dates you want from the following:

  - lastHour—valid for hourly reports only
  - last24Hours—valid for hourly or daily reports
  - last7Days—valid for all reports
  - last30Days—valid for daily reports only
  - last90Days—valid for daily reports only
- startDate <msecs>—Report start date in milliseconds since 1970. If this parameter is provided, the endDate must be provided.
- endDate <msecs>—Report end date in milliseconds since 1970. The endDate must be greater than the start date. If this parameter is provided, the startDate must be provided.
- outputType <string>—Output type: CSV, TABLE, or GRAPH.
- topStatSel <string>—Default data sorting. The provided string must match the string that appears in the GUI drop down.
- screenWidth <integer>—The screen width. The default is 1000 pixels.
- screenHeight <integer>—The screen height. The default is 700 pixels.
- clientOffSetTime <msecs>—The offset for the the client time zone. If not provided, the report start and end dates use the server time zone.
- fullScreen <integer>—The graph index to show as full screen. This is a 0 index number.
- hideButtons<boolean>—Hides the graph Zoom, Graph Style, and Export options.
- hideTitle <boolean>—Hides the graph title.
- hideNavigator <boolean>—Hides the navigator in leaf and full screen graphs.
- hideLegend <boolean>—Hides the legend in graphs.
- hideDateString <boolean>—Hides the date in graphs.


# Managing Prime Performance Manager Using the NBAPI

Prime Performance Manager north bound API (NBAPI) allows you to perform some device, threshold, polling group, and user management functions including adding, updating, and deleting devices and thresholds, and adding, deleting and editing users. NBAPI information is provided at the following locations:

- Address—http://*ppm_gateway*:4440/nbapi/admin
- WSDL—http://*ppm_gateway*:4440/nbapi/admin?wsdl
- Implementation class—com.cisco.mwg.sgm.nbapi.server.AdminAPIImpl
- Project WSDL—/opt/CSCOppm-gw/tomcat/webapps/nbapi/WEB-INF/wsdl/AdminAPI.wsdl

**Note** If Prime Performance Manager is integrated into the Cisco Prime Carrier Management suite, the interface is https://*ppm_gateway*:4440/nbapi/event.

**Note** If user access is enabled, SOAP requests must include a Prime Performance Manager user name and password through the HTTP Basic Authorization Header. The user must have sufficient permissions to perform the requested SOAP action.

Examples of NBAPI methods are provided in the following topics:

# Device Management Methods

Device management methods include:



## addDevice

Examples of the NBAPI addDevice method are provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addDevice>
         <ipaddress>?</ipaddress>
         <name>?</name>
         <syncName>?</syncName>
         <webPort>?</webPort>
         <timeZone>?</timeZone>
         <location>?</location>
         <isManaged>?</isManaged>
         <isSendingAlarms>?</isSendingAlarms>
         <reportPolicy>?</reportPolicy>
        <pollingCollector>?</pollingCollector>
         <pollingGroup>?</pollingGroup>
        <unit>?</unit>
         <snmpVersion>?</snmpVersion>
         <snmpReadCommunity>?</snmpReadCommunity>
         <snmpAuthProtocol>?</snmpAuthProtocol>
         <snmpAuthPwd>?</snmpAuthPwd>
         <snmpPrivProtocol>?</snmpPrivProtocol>
         <snmpPrivPwd>?</snmpPrivPwd>
         <snmpUserName>?</snmpUserName>
         <loginProtocol>?</loginProtocol>
         <loginPortNumber>?</loginPortNumber>
         <loginUserName>?</loginUserName>
         <loginPassword>?</loginPassword>
         <loginEnableUserName>?</loginEnableUserName>
         <loginEnablePassword>?</loginEnablePassword>
         <loginSubsystem>?</loginSubsystem>
         <loginSecondaryType>?</loginSecondaryType>
         <loginClientAuthType>?</loginClientAuthType>
         <loginClientAuthPublicKey>?</loginClientAuthPublicKey>
         <loginClientAuthPrivateKey>?</loginClientAuthPrivateKey>
      </ppm:addDevice>
   </soapenv:Body>
</soapenv:Envelope>
```

An addDevice example for a device with SNMP v3 and Telnet credentials:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
    <soapenv:Header/>
```

```
    <soapenv:Body>
       <ppm:addDevice>
          <ipaddress>172.18.53.231</ipaddress>
          <name>nms7606e</name>
          <syncName>nms-sync-name</syncName>
          <webPort>1212</webPort>
          <timeZone>US/Eastern</timeZone>
          <location>RTP-3rd Floor CISCO Lab</location>
          <isManaged>true</isManaged>
          <isSendingAlarms>FALSE</isSendingAlarms>
          <reportPolicy>cisco7600</reportPolicy>
          <pollingGroup>c7600-group</pollingGroup>
          <pollingCollector></pollingCollector>
          <snmpVersion>2c</snmpVersion>
          <snmpReadCommunity>public</snmpReadCommunity>
          <snmpAuthProtocol>md5</snmpAuthProtocol>
          <snmpAuthPwd>cisco123</snmpAuthPwd>
          <snmpPrivProtocol>des</snmpPrivProtocol>
          <snmpPrivPwd>cisco123</snmpPrivPwd>
          <snmpUserName>adminuser</snmpUserName>
          <loginProtocol>Telnet</loginProtocol>
          <loginPortNumber>23</loginPortNumber>
          <loginUserName>admin</loginUserName>
          <loginPassword>lab</loginPassword>
          <loginEnableUserName>admin</loginEnableUserName>
          <loginEnablePassword>lab</loginEnablePassword>
          <loginSubsystem></loginSubsystem>
          <loginSecondaryType>Enable</loginSecondaryType>
          <loginClientAuthType>Password</loginClientAuthType>
          <loginClientAuthPublicKey></loginClientAuthPublicKey>
          <loginClientAuthPrivateKey></loginClientAuthPrivateKey>
          <unit>10.81.82.92</unit>
       </ppm:addDevice>
    </soapenv:Body>
 </soapenv:Envelope>
```

An addDevice example for a non-SNMP enabled device: QEMU/KVM Hypervisor

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addDevice>
         <ipaddress>10.81.83.114</ipaddress>
         <name>ucs-alpha-200a</name>
         <syncName></syncName>
         <webPort></webPort>
         <timeZone>Hongkong</timeZone>
         <location></location>
         <isManaged>true</isManaged>
         <isSendingAlarms>true</isSendingAlarms>
         <reportPolicy></reportPolicy>
         <pollingGroup></pollingGroup>
         <pollingCollector>Hypervisor,OpenStack</pollingCollector>
         <snmpVersion></snmpVersion>
         <snmpReadCommunity>public</snmpReadCommunity>
         <snmpAuthProtocol></snmpAuthProtocol>
         <snmpAuthPwd></snmpAuthPwd>
         <snmpPrivProtocol></snmpPrivProtocol>
         <snmpPrivPwd></snmpPrivPwd>
         <snmpUserName></snmpUserName>
         <loginProtocol>KVM_TLS</loginProtocol>
         <loginPortNumber>16514</loginPortNumber>
         <loginUserName>kvmuser</loginUserName>
```

```
            <loginPassword>cisco123</loginPassword>
            <loginEnableUserName></loginEnableUserName>
            <loginEnablePassword></loginEnablePassword>
            <loginSubsystem></loginSubsystem>
            <loginSecondaryType>Enable</loginSecondaryType>
            <loginClientAuthType>Password</loginClientAuthType>
            <loginClientAuthPublicKey></loginClientAuthPublicKey>
            <loginClientAuthPrivateKey></loginClientAuthPrivateKey>
            <unit></unit>
         </ppm:addDevice>
      </soapenv:Body>
   </soapenv:Envelope>
```

# updateDevice

Examples of the NBAPI updateDevice method are provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updateDevice>
         <ipaddress>?</ipaddress>
         <name>?</name>
         <syncName>?</syncName>
         <webPort>?</webPort>
         <timeZone>?</timeZone>
         <location>?</location>
         <isManaged>?</isManaged>
         <isSendingAlarms>?</isSendingAlarms>
        <pollingCollector>?</pollingCollector
         <reportPolicy>?</reportPolicy>
         <pollingGroup>?</pollingGroup>
         <snmpVersion>?</snmpVersion>
         <snmpReadCommunity>?</snmpReadCommunity>
         <snmpAuthProtocol>?</snmpAuthProtocol>
         <snmpAuthPwd>?</snmpAuthPwd>
         <snmpPrivProtocol>?</snmpPrivProtocol>
         <snmpPrivPwd>?</snmpPrivPwd>
         <snmpUserName>?</snmpUserName>
         <loginProtocol>?</loginProtocol>
         <loginPortNumber>?</loginPortNumber>
         <loginUserName>?</loginUserName>
         <loginPassword>?</loginPassword>
        <loginDelete>?</loginDelete
         <loginEnableUserName>?</loginEnableUserName>
         <loginEnablePassword>?</loginEnablePassword>
         <loginSubsystem>?</loginSubsystem>
         <loginSecondaryType>?</loginSecondaryType>
         <loginClientAuthType>?</loginClientAuthType>
         <loginClientAuthPublicKey>?</loginClientAuthPublicKey>
         <loginClientAuthPrivateKey>?</loginClientAuthPrivateKey>
      </ppm:updateDevice>
   </soapenv:Body>
</soapenv:Envelope>
&
```

An updateDevice example with SSH_V2 credentials and public and private keys:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <ppm:updateDevice>
         <ipaddress>10.74.125.169</ipaddress>
         <name></name>
         <syncName></syncName>
         <webPort></webPort>
         <timeZone></timeZone>
         <location></location>
         <isManaged>true</isManaged>
         <isSendingAlarms>true</isSendingAlarms>
         <reportPolicy>Default</reportPolicy>
         <pollingGroup></pollingGroup>
         <pollingCollector></pollingCollector>
         <snmpVersion></snmpVersion>
         <snmpReadCommunity>public</snmpReadCommunity>
         <snmpAuthProtocol></snmpAuthProtocol>
         <snmpAuthPwd></snmpAuthPwd>
         <snmpPrivProtocol></snmpPrivProtocol>
         <snmpPrivPwd></snmpPrivPwd>
         <snmpUserName></snmpUserName>
         <loginProtocol>SSH_V2</loginProtocol>
         <loginPortNumber>22</loginPortNumber>
         <loginUserName>ubuntu</loginUserName>
         <loginPassword></loginPassword>
         <loginEnableUserName></loginEnableUserName>
         <loginEnablePassword></loginEnablePassword>
         <loginSubsystem></loginSubsystem>
         <loginSecondaryType>Enable</loginSecondaryType>
         <loginClientAuthType>PublicKey</loginClientAuthType>
         <loginClientAuthPublicKey>ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCmWdPAldSGJYUG+n1IVaez+XAQDEza1HdI0fgozhhxM0bSZfLLQUZmReolLd
FdIW/pNBca8nCKYEJaUgitmr3B7htzcK2u92EF7xqu+UJkd0/7yCbq6Rn2nt1rB4gknGvy5MmBDijmIWyRGCgMpGZp
WcH9QK7Toyua4/gZGeFnW8wAK9PMD6jkdgM8kpyuo4ypTLvPrJI3ZTCwAXeYM3B3j11JVY45pawCTogl3bjYrNGUgE
IaYB4ZiVO4chUFbuQL251w79vOwwjBQ8ec93V ubuntu@crdc-c210-169</loginClientAuthPublicKey>
         <loginClientAuthPrivateKey>---BEGIN RSA PRIVATE
KEY-MIIEowIBAAKCAQEAplnTwJXUhiWFBvp9SFWns/lwEAxM2tR3SNH4KM4YcTNG0mXyy0FGZkXqJS3RXSFv4M0H6G
SyUWwnre8rsQbfWQxo3A5OpJcPpfxLgxv4Kcrm+U/nZlW3MQKBgHBhJaCM7XFbS22UrZLngGK6th/A0ui+2WmGQAl2
cSpnNX3QdPxOTjgz2rEY8OfPMiCsJynI75npq0NKvTM+wk0/ixT9RnSnVyFh04AVgXw80qx1rwmCxupuuHEJyX+7cM
IPelpSsjCLROgu0F5ggm3blfSQ3rt+ThxB7Ek0d1jR-END RSA PRIVATE
KEY---</loginClientAuthPrivateKey>
         <loginDelete></loginDelete>
      </ppm:updateDevice>
   </soapenv:Body>
</soapenv:Envelope>
```

# deleteDevice

An example of the NBAPI deleteDevice method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:deleteDevice>
         <ipaddress>172.18.156.168</ipaddress>
      </ppm:deleteDevice>
   </soapenv:Body>
</soapenv:Envelope>
```

# Add, Update, Delete Device API Tag Descriptions

Descriptions of the add, update, and delete device attributes are provided below. Many of the following attributes are visible in Prime Performance Manager GUI, when you select the device and display the Details tab.

- ipaddress—Required. IP address. The IP address used to discover the device in Prime Performance Manager.
- name—Text. The device display name. This can be different from system name (name set on the router and returned, using the SNMP variable sysName). The name must be unique in Prime Performance Manager.
- syncName—Text. Generally used in Prime Network context. It is the device name (or business tag, if defined) as it appears in Prime Network. syncName must be unique in Prime Performance Manager.
- webPort—Number. The specific port number to launch the device home page.
- timeZone—Text. Time zone of the device. Valid examples are US/Eastern, EST, Asia/Shanghai, Etc/GMT-4, Australia/Sydney, Europe/Moscow, etc. The expanded form is preferred for accuracy.
- location—Text. Description of the device physical location.
- isManaged—true or false. If the device must be managed or unmanaged state. Only in managed state can Prime Performance Manager gather statistics from device.
- isSendingAlarms—true or false. If this devices alarms need to be sent to other NB interfaces.
- pollingCollector—Indicate the collectors to be used for discovering the device. Valid options are collectd, UCS-CIMC, AVI, UCSM, OpenStack, gmond, WMI, Hypervisor, SMI, Default. Prime Performance Manager uses SNMP as the primary protocol to discover a device and assign the device type based on the sysObjectID value. If SNMP should not be used for discovery or if other collectors are available for this device, indicate the collector names separated by commas without any space in-between.

**Note** Prerequisite is to set the MULTI_COLLECTOR_ENABLED property to true in Server.properties file on both gateway and all the connected units. A restart of the gateway is required for the property change to take effect.

### Policy and Polling Groups

Available groups can be seen in the Prime Performance Manager GUI in Report/Group Policies and Polling Group Editor sections.

- reportPolicy—Text. Name of the policy the device must use. reportPolicy indicates which reports run for this device. After discovery, Prime Performance Manager assigns report policy based on the device type. If you must override the default operation and assign a specific report policy, send the report policy name.

  **Note** reportPolicy should exist in Prime Performance Manager.

- pollingGroup—Text. Name of the polling group this device needs to use. A pollingGroup describes values like Polling Interval, Device SNMP Timeout value, SNMP Retries used for the device when Prime Performance Manager polls the device. After discovering Prime Performance Manager assigns polling group based on the device type. If you must override the default operation and assign a specific polling group send the polling group name.

**Note** pollingGroup should exist in Prime Performance Manager.

#### SNMP Information

The following tags describe the SNMP attributes to be used when polling the device through SNMP. The SNMP Editor in the Prime Performance Manager GUI displays these values.

- snmpVersion—Options. SNMP Version to be use. Valid options are 1, 2c, 3
- snmpReadCommunity—Text. Read Community string used for SNMP version 2c.
- snmpAuthProtocol—Options. SNMP version 3 - Authentication Protocol. Valid options are md5, sha
- snmpAuthPwd—Text. SNMP version 3 authentication password
- snmpPrivProtocol—Options. SNMP version 3 privacy protocol. Valid options are des, 3des, aes128
- snmpPrivPwd—Text. SNMP version 3 privacy password
- snmpUserName—Text. SNMP version 3 username.

#### Some Rules for SNMP

- For SNMP collection—At a minimum either snmpReadCommunity or snmpUserName is required for polling.
- snmpUserName is required when other SNMP version 3 attributes are provided.
- Valid SNMP version 3 combinations are:
  - snmpUserName only (NoAuthNoPriv)
  - snmpUserName and snmpAuthProtocol (AuthNoPriv)
  - snmpUserName, snmpAuthProtocol, snmpPrivProtocol (AuthPriv)
- snmpAuthPwd is required when snmpAuthProtocol is provided.
- snmpPrivPwd is required when snmpPrivProtocol is provided.
- While updating, providing all the mandatory attributes instead of updating a single attribute is recommended.

#### Credential Information

The following tags describe the credentials Prime Performance Manager will use while accessing the device through different collectors. You can view the added credentials through the Network > Credentials Editor window.

- loginProtocol— Required if credentials are provided. Options. The transport protocol to be used to communicate with the device. Valid options are ESXi_HTTPS, HyperV_HTTPS, HTTP_BULK, HTTP, SSH_V1, WSMA_SSH, SSH_V2, KVM_SSH, KVM_TLS, vCenter_HTTP, WSMA_HTTPS, XEN_SSH, HyperV_HTTP, HTTPS, XML_SSL, SMI_HTTPS, vCenter_HTTPS, WMI_HTTP, JMX, PNSC_HTTPS, collectd_SSH, WMI_HTTPS, GMOND_SOCKET, Telnet, XEN_TLS, WSMA_HTTP, AVI_HTTPS, XML_Telnet, WSMA_TLS, ESXi_HTTP, ULS_HTTP
  - Telnet—Telnet.
  - SSHv1—SSH Version 1.
  - SSHv2—SSH Version 2.

  - WSMA_SSH—Web Services Management Agent over SSHv2. WSMA is an infrastructure framework that allows external applications to monitor and control Cisco devices. WSMA uses transports such as SSH, HTTP, and HTTPS to access a set of Web Services agents residing on the Cisco device.
  - collectd_SSH—A daemon that collects, transfers, and stores performance data.
  - HTTP—HyperText Transfer Protocol.
  - HTTPS—Secure HTTP.
  - HTTP_BULK—Bulk statistics through HTTP.
  - WMI_HTTP—Windows Management Instrumentation over HTTP.
  - WMI_HTTPS—Windows Management Instrumentation HTTPS.
  - SMI_HTTPS—Storage Management Initiative over HTTPS.
  - ULS_HTTP—Allows Prime Performance Manager to perform Small Cell upload server HTTP credential verification including subsystem, username, password, and credential parameters. Beyond that, ULS_HTTP is identical to HTTP protocol.
  - vCenter_HTTPS—VMware vCenter server over HTTPS.
  - ESXi_HTTP—VMware ESXi embedded bare metal hypervisor over HTTP.
  - ESXi_HTTPS—VMware ESXi embedded bare metal hypervisor over HTTPS
  - XEN_TLS—Xen hypervisor over Transport Layer Security (TLS) protocol.
  - KVM_TLS—Linux Kernal-based Virtual Machine (KVM) over TLS.
  - HyperV_HTTP—Microsoft HyperV server over HTTP.
  - HyperV_HTTPS—Microsoft HyperV server over HTTPS.
  - JMX—Java Management Extensions. Collects statistics from Java processes running on various servers.
  - PNSC_HTTPS—Cisco Prime Network Services Controller secure HTTP connection.
  - GMOND_SOCKET—Ganglia Monitoring Daemon (gmond) socket.

**Note** For more information about different transport protocol refer Prime Performance Manager 1.7 User Guide “Adding Device Credentials for Other Protocols” section.

- loginPortNumber—Required if credentials are provided. Number. The port number used for the transport protocol.
- loginUserName—Optional depending on device configuration. Text. Login User name needed for the device.
- loginPassword—Optional depending on device configuration. Text. Login password for the device.
- loginEnableUserName—Optional depending on device configuration. Text. If device requires enable / secondary user name use this tag. Mostly used for Cisco IOS devices.
- loginEnablePassword—Optional depending on device configuration. Text. The enable / secondary user password for the device.
- loginSubsystem—Optional. Text. The subsystem used by transport protocol. If the subsystem is defined on the device, enter it here.

**Note** NOTE: A blank string is the default subsystem for SSH. The default subsystem for WSMA is "wsma".

- loginSecondaryType—Required if credentials are provided. Options. Indicates how the secondary user and password (loginEnableUserName, loginEnablePassword) should be processed. Valid options are Enable, SecondLogin. Option Enable executes the Cisco IOS enable command, while the SecondLogin option uses the same fields as secondary login to the device.
- loginClientAuthType—Required if credentials are provided. Options. Valid options are Password or PublicKey.

**Note** NOTE: For SSHv2 or HTTPS transport protocol use PublicKey option and pass the authentication keys in the loginClientAuthPublicKey and loginClientAuthPrivateKey tags. By default, Prime Performance Manager authenticates itself to the device using the User Name and Password entries.

- loginClientAuthPublicKey—Required for PublicKey loginClientAuthType. SSHv2 Authentication Public key.
- loginClientAuthPrivateKey—Required for PublicKey loginClientAuthType. SSHv2 Authentication Private key.
- loginDelete—true or false. A true indicates the login credential entered in above login tags will need to be deleted for this device. This option can be used to remove unwanted device credentials. This tag is available only for updateDevice API.

**Note** To add multiple login credentials for the same device use the updateDevice API.

#### Unit Assignment

- unit—Optional. IP or hostname of the Unit server. By default, Prime Performance Manager assigns units to discovered devices automatically. If you want to assign the device to a specific unit use this field to indicate the unit server.

  This tag is available only for addDevice API.

**Note** NOTE: true or false values are case-insensitive. So you can also send as True or FALSE.

**Note** NOTE: Option values are case sensitive.

## getDeviceInfo

An example of the NBAPI getDeviceInfo method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getDeviceInfo>
         <DeviceIdList>
```

```
            <!--1 or more repetitions:-->
            <name>?</name>
         </DeviceIdList>
      </ppm:getDeviceInfo>
   </soapenv:Body>
</soapenv:Envelope>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getDeviceInfoResponse xmlns:ns2="http://cisco.com/ppm">
         <DeviceInfoList>
            <Device>
               <ipaddress>1.2.3.4</ipaddress>
               <name>ppm-vm1234.cisco.com</name>
               <webPort>23</webPort>
               <timeZone>Eastern Standard Time</timeZone>
               <isManaged>true</isManaged>
               <isSendingAlarms>true</isSendingAlarms>
               <reportPolicy>Default</reportPolicy>
               <pollingGroup>Linux</pollingGroup>
               <extPropertiesList>
                  <ExtProperty name="AlarmSeverity">
                     <Value>Critical</Value>
                  </ExtProperty>
                  <ExtProperty name="DeviceType">
                     <Value>Linux</Value>
                  </ExtProperty>
                  <ExtProperty name="DiscoveredTimestamp">
                     <Value>2014-06-13T14:32:07.257-04:00</Value>
                  </ExtProperty>
                  <ExtProperty name="DiscoverySource">
                     <Value>PPM</Value>
                  </ExtProperty>
                  <ExtProperty name="GNE">
                     <Value/>
                  </ExtProperty>
                  <ExtProperty name="LastCapabilityPollTimestamp">
                     <Value>2014-06-18T03:47:22.581-04:00</Value>
                  </ExtProperty>
                  <ExtProperty name="LastPollTimestamp">
                     <Value>2014-06-18T10:26:32.914-04:00</Value>
                  </ExtProperty>
                  <ExtProperty name="ParsedVersion">
                     <Value>5.3.2.2</Value>
                  </ExtProperty>
                  <ExtProperty name="RebootReason">
                     <Value/>
                  </ExtProperty>
                  <ExtProperty name="SerialNumber">
                     <Value/>
                  </ExtProperty>
                  <ExtProperty name="Slot">
                     <Value>-1</Value>
                  </ExtProperty>
                  <ExtProperty name="State">
                     <Value>Active</Value>
                  </ExtProperty>
                  <ExtProperty name="StateReason">
                     <Value>None</Value>
                  </ExtProperty>
                  <ExtProperty name="StateTimestamp">
                     <Value>2014-06-17T14:17:23.409-04:00</Value>
                  </ExtProperty>
                   <ExtProperty name="MaintenanceStartTime">
```

```
                        <Value>2014-06-17T14:17:23.409-04:00</Value>
                     </ExtProperty>
                      <ExtProperty name="MaintenanceEndTime">
                        <Value>2014-06-27T12:19:09.409-04:00</Value>
                     </ExtProperty>
                      <ExtProperty name="MaintenanceStatus">
                        <Value>Disabled</Value>
                     </ExtProperty>
                     <ExtProperty name="SysContact">
                        <Value>jkinder</Value>
                     </ExtProperty>
                     <ExtProperty name="SysDescr">
                       <Value>Linux ppm-vm1234 2.6.18-194.8.1.el5 #1 SMP Wed Jun 23 10:52:51
EDT 2010 x86_64</Value>
                     </ExtProperty>
                     <ExtProperty name="SysLocation">
                        <Value>RTP6P</Value>
                     </ExtProperty>
                     <ExtProperty name="SysName">
                        <Value>ppm-vm1234</Value>
                     </ExtProperty>
                     <ExtProperty name="SysUpTime">
                        <Value>P30DT17H20M24.990S</Value>
                     </ExtProperty>
                     <ExtProperty name="UniqueId">
                        <Value>1.2.3.4@@-1</Value>
                     </ExtProperty>
                  </extPropertiesList>
               </Device>
            </DeviceInfoList>
         </ns2:getDeviceInfoResponse>
      </S:Body>
</S:Envelope>

&&&ADD MaintenanceEndTime & & MaintenanceState
```

# Threshold Management Methods

NBAPI threshold method examples are provided below. If you do not specify a field, the code takes the default value. However, if you explicitly leave the field empty, the code clears the default value unless the field is required.

The sample responses are for traditional number thresholds, so they don't show the <xxxOperator> and <xxxTestValue> elements in the KPI section because those are null. However, they would be present for a string threshold. The valid values for xxxOperator are the same as for report filtering: Equals, Does Not Equal, Contains, Does Not Contain, Begins With, Ends With. The valid xxxTestValuevalues for are anything that can go in xml, and there can be any number of them. The Operators starting with "Does Not" pass if the positive version does not match any of the test values. The other tests match if any test value matches. You cannot create or edit string thresholds with addThreshold or editThreshold. You must

use the addThresholdInfoList or editThresholdInfoList methods. Also, abate tests are optional and not recommended for string thresholds. If they are omitted, the abateOccurs can still be specified. The threshold will abate when the onset test fails that number of times.

A <mailFrom> element is included in all request and response WSDL that has a <mailTo> element. The <mailFrom> element occurs first. On the CLI side, the mailFrom can be specified in addThreshold and getFilteredThresholdInfo through an -f option. The addThresholdInfoList and editThresholdInfoList methods that take an xml file with the -u option like the Probe equivalents.

For REST reports, you can display threshold API parameters by choosing **Help > Reports > Threshold API Parameters** in the Prime Performance Manager GUI. The page includes the threshold reportKey and kpiReport parameters. Metadata is added for each column including dataType, columnName, and kpiName. (columnName and kpiName are empty for all non-thresholdable columns.)

You can also run the following command to generate a list of reports available from the REST interface:

```
/opt/CSCOppm-gw/bin/sgmGenReportAutoDoc.sh genthreshold <file_or_folder_name>
```

Each report will include the following parameters:

- REST URL
- name
- reportKey
- kpiReport
- A set of thresholdable columns, each with the following parameters:
  - reportHeader
  - columnName
  - kpiName

If file or folder name ends in .xml, and is not an existing directory, the XML is generated in that file. If file or folder name does not end in .xml or is an existing directory, the XML is generated in a ThresholdParameters.xml file in that directory.

Threshold management methods are provided in the following topics:



# addThreshold

An example of the NBAPI addThreshold method is shown below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addThreshold>
         <userThresholdName>?</userThresholdName>
         <columnName>?</columnName>
         <reportKey>?</reportKey>
         <kpiName>?</kpiName>
         <kpiReport>?</kpiReport>
         <rising>?</rising>
         <interval>?</interval>
         <baselineEnabled>?</baselineEnabled>
         <baselineOperation>?</baselineOperation>
         <baselineWindowSize>?</baselineWindowSize>
         <scope>?</scope>
         <description>?</description>
         <enabled>?</enabled>
         <continuousAlarmEnabled>?</continuousAlarmEnabled>
         <alarmType>?</alarmType>
         <probableCause>?</probableCause>
         <alarmNature>?</alarmNature>
         <alarmScript>?</alarmScript>
         <mailFrom>?</mailFrom>
         <mailTo>?</mailTo>
         <mailSubject>?</mailSubject>
         <msgText>?</msgText>
         <Monday>?</Monday>
         <Tuesday>?</Tuesday>
         <Wednesday>?</Wednesday>
         <Thursday>?</Thursday>
         <Friday>?</Friday>
         <Saturday>?</Saturday>
         <Sunday>?</Sunday>
         <beginTime>?</beginTime>
         <endTime>?</endTime>
         <criticalOnset>?</criticalOnset>
         <criticalOnsetOccurs>?</criticalOnsetOccurs>
         <criticalAbate>?</criticalAbate>
         <criticalAbateOccurs>?</criticalAbateOccurs>
         <criticalPolicyOverride>?</criticalPolicyOverride>
         <majorOnset>?</majorOnset>
         <majorOnsetOccurs>?</majorOnsetOccurs>
         <majorAbate>?</majorAbate>
         <majorAbateOccurs>?</majorAbateOccurs>
         <majorPolicyOverride>?</majorPolicyOverride>
         <minorOnset>?</minorOnset>
         <minorOnsetOccurs>?</minorOnsetOccurs>
         <minorAbate>?</minorAbate>
         <minorAbateOccurs>?</minorAbateOccurs>
         <minorPolicyOverride>?</minorPolicyOverride>
         <warningOnset>?</warningOnset>
         <warningOnsetOccurs>?</warningOnsetOccurs>
         <warningAbate>?</warningAbate>
         <warningAbateOccurs>?</warningAbateOccurs>
         <warningPolicyOverride>?</warningPolicyOverride>
         <informationalOnset>?</informationalOnset>
         <informationalOnsetOccurs>?</informationalOnsetOccurs>
         <informationalAbate>?</informationalAbate>
         <informationalAbateOccurs>?</informationalAbateOccurs>
         <informationalPolicyOverride>?</informationalPolicyOverride>
         <tenants>?</tenants>
         <excludedScopeKeys>?</excludedScopeKeys>
      </ppm:addThreshold>
```

```
   </soapenv:Body>
</soapenv:Envelope>
```

# addThresholdInfoList

An example of the NBAPI addThresholdInfoList method is shown below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addThresholdInfoList>
         <thresholdInfoList>
            <!--1 or more repetitions:-->
            <ThresholdInfo>
               <userThresholdName>?</userThresholdName>
               <columnName>?</columnName>
               <reportKey>?</reportKey>
               <kpiName>?</kpiName>
               <kpiReport>?</kpiReport>
               <rising>?</rising>
               <interval>?</interval>
               <compoundOperation>?</compoundOperation>
               <baselineEnabled>?</baselineEnabled>
               <baselineOperation>?</baselineOperation>
               <baselineWindowSize>?</baselineWindowSize>
               <scope>?</scope>
               <description>?</description>
               <enabled>?</enabled>
               <continuousAlarmEnabled>?</continuousAlarmEnabled>
               <alarmType>?</alarmType>
               <probableCause>?</probableCause>
               <alarmNature>?</alarmNature>
               <alarmScript>?</alarmScript>
               <mailFrom>?</mailFrom>
               <mailTo>?</mailTo>
               <mailSubject>?</mailSubject>
               <msgText>?</msgText>
               <Monday>?</Monday>
               <Tuesday>?</Tuesday>
               <Wednesday>?</Wednesday>
               <Thursday>?</Thursday>
               <Friday>?</Friday>
               <Saturday>?</Saturday>
               <Sunday>?</Sunday>
               <beginTime>?</beginTime>
               <endTime>?</endTime>
               <criticalOnset>?</criticalOnset>
               <criticalOnsetOccurs>?</criticalOnsetOccurs>
               <criticalAbate>?</criticalAbate>
               <criticalAbateOccurs>?</criticalAbateOccurs>
               <criticalPolicyOverride>?</criticalPolicyOverride>
               <majorOnset>?</majorOnset>
               <majorOnsetOccurs>?</majorOnsetOccurs>
               <majorAbate>?</majorAbate>
               <majorAbateOccurs>?</majorAbateOccurs>
               <majorPolicyOverride>?</majorPolicyOverride>
               <minorOnset>?</minorOnset>
               <minorOnsetOccurs>?</minorOnsetOccurs>
               <minorAbate>?</minorAbate>
               <minorAbateOccurs>?</minorAbateOccurs>
               <minorPolicyOverride>?</minorPolicyOverride>
```

```
<warningOnset>?</warningOnset>
<warningOnsetOccurs>?</warningOnsetOccurs>
<warningAbate>?</warningAbate>
<warningAbateOccurs>?</warningAbateOccurs>
<warningPolicyOverride>?</warningPolicyOverride>
<informationalOnset>?</informationalOnset>
<informationalOnsetOccurs>?</informationalOnsetOccurs>
<informationalAbate>?</informationalAbate>
<informationalAbateOccurs>?</informationalAbateOccurs>
<informationalPolicyOverride>?</informationalPolicyOverride>
<kpis>
   <!--1 or more repetitions:-->
   <KPIInfo>
      <columnName>?</columnName>
      <reportKey>?</reportKey>
      <kpiName>?</kpiName>
      <kpiReport>?</kpiReport>
      <rising>?</rising>
      <baselineEnabled>?</baselineEnabled>
      <baselineOperation>?</baselineOperation>
      <baselineWindowSize>?</baselineWindowSize>
      <scope>?</scope>
      <!--Optional:-->
      <criticalOnset>?</criticalOnset>
      <!--Optional:-->
      <criticalOnsetOperator>?</criticalOnsetOperator>
      <!--Zero or more repetitions:-->
      <criticalOnsetTestValue>?</criticalOnsetTestValue>
      <!--Optional:-->
      <criticalAbate>?</criticalAbate>
      <!--Optional:-->
      <criticalAbateOperator>?</criticalAbateOperator>
      <!--Zero or more repetitions:-->
      <criticalAbateTestValue>?</criticalAbateTestValue>
      <!--Optional:-->
      <majorOnset>?</majorOnset>
      <!--Optional:-->
      <majorOnsetOperator>?</majorOnsetOperator>
      <!--Zero or more repetitions:-->
      <majorOnsetTestValue>?</majorOnsetTestValue>
      <!--Optional:-->
      <majorAbate>?</majorAbate>
      <!--Optional:-->
      <majorAbateOperator>?</majorAbateOperator>
      <!--Zero or more repetitions:-->
      <majorAbateTestValue>?</majorAbateTestValue>
      <!--Optional:-->
      <minorOnset>?</minorOnset>
      <!--Optional:-->
      <minorOnsetOperator>?</minorOnsetOperator>
      <!--Zero or more repetitions:-->
      <minorOnsetTestValue>?</minorOnsetTestValue>
      <!--Optional:-->
      <minorAbate>?</minorAbate>
      <!--Optional:-->
      <minorAbateOperator>?</minorAbateOperator>
      <!--Zero or more repetitions:-->
      <minorAbateTestValue>?</minorAbateTestValue>
      <!--Optional:-->
      <warningOnset>?</warningOnset>
      <!--Optional:-->
      <warningOnsetOperator>?</warningOnsetOperator>
      <!--Zero or more repetitions:-->
      <warningOnsetTestValue>?</warningOnsetTestValue>
```

```
                        <!--Optional:-->
                        <warningAbate>?</warningAbate>
                        <!--Optional:-->
                        <warningAbateOperator>?</warningAbateOperator>
                        <!--Zero or more repetitions:-->
                        <warningAbateTestValue>?</warningAbateTestValue>
                        <!--Optional:-->
                        <informationalOnset>?</informationalOnset>
                        <!--Optional:-->
                        <informationalOnsetOperator>?</informationalOnsetOperator>
                        <!--Zero or more repetitions:-->
                        <informationalOnsetTestValue>?</informationalOnsetTestValue>
                        <!--Optional:-->
                        <informationalAbate>?</informationalAbate>
                        <!--Optional:-->
                        <informationalAbateOperator>?</informationalAbateOperator>
                        <!--Zero or more repetitions:-->
                        <informationalAbateTestValue>?</informationalAbateTestValue>
                     </KPIInfo>
                  </kpis>
                  <tenants>
                     <!--1 or more repetitions:-->
                     <Tenant>?</Tenant>
                  </tenants>
                  <excludedScopeKeys>?</excludedScopeKeys>
               </ThresholdInfo>
            </thresholdInfoList>
         </ppm:addThresholdInfoList>
      </soapenv:Body>
   </soapenv:Envelope>
```

# editThreshold

An example of the NBAPI editThreshold method is provided below.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:editThreshold>
         <userThresholdName>?</userThresholdName>
         <rising>?</rising>
         <interval>?</interval>
         <baselineEnabled>?</baselineEnabled>
         <baselineOperation>?</baselineOperation>
         <baselineWindowSize>?</baselineWindowSize>
         <scope>?</scope>
         <description>?</description>
         <enabled>?</enabled>
         <continuousAlarmEnabled>?</continuousAlarmEnabled>
         <alarmType>?</alarmType>
         <probableCause>?</probableCause>
         <alarmNature>?</alarmNature>
         <alarmScript>?</alarmScript>
         <mailFrom>?</mailFrom>
         <mailTo>?</mailTo>
         <mailSubject>?</mailSubject>
         <msgText>?</msgText>
         <Monday>?</Monday>
         <Tuesday>?</Tuesday>
         <Wednesday>?</Wednesday>
         <Thursday>?</Thursday>
```

```
            <Friday>?</Friday>
            <Saturday>?</Saturday>
            <Sunday>?</Sunday>
            <beginTime>?</beginTime>
            <endTime>?</endTime>
            <criticalOnset>?</criticalOnset>
            <criticalOnsetOccurs>?</criticalOnsetOccurs>
            <criticalAbate>?</criticalAbate>
            <criticalAbateOccurs>?</criticalAbateOccurs>
            <criticalPolicyOverride>?</criticalPolicyOverride>
            <majorOnset>?</majorOnset>
            <majorOnsetOccurs>?</majorOnsetOccurs>
            <majorAbate>?</majorAbate>
            <majorAbateOccurs>?</majorAbateOccurs>
            <majorPolicyOverride>?</majorPolicyOverride>
            <minorOnset>?</minorOnset>
            <minorOnsetOccurs>?</minorOnsetOccurs>
            <minorAbate>?</minorAbate>
            <minorAbateOccurs>?</minorAbateOccurs>
            <minorPolicyOverride>?</minorPolicyOverride>
            <warningOnset>?</warningOnset>
            <warningOnsetOccurs>?</warningOnsetOccurs>
            <warningAbate>?</warningAbate>
            <warningAbateOccurs>?</warningAbateOccurs>
            <warningPolicyOverride>?</warningPolicyOverride>
            <informationalOnset>?</informationalOnset>
            <informationalOnsetOccurs>?</informationalOnsetOccurs>
            <informationalAbate>?</informationalAbate>
            <informationalAbateOccurs>?</informationalAbateOccurs>
            <informationalPolicyOverride>?</informationalPolicyOverride>
            <tenants>?</tenants>
            <excludedScopeKeys>?</excludedScopeKeys>
         </ppm:editThreshold>
      </soapenv:Body>
   </soapenv:Envelope>
```

# editThresholdInfoList

An example of the NBAPI editThreshold InfoList method is shown below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:editThresholdInfoList>
         <thresholdInfoList>
            <!--1 or more repetitions:-->
            <ThresholdInfo>
               <userThresholdName>?</userThresholdName>
               <columnName>?</columnName>
               <reportKey>?</reportKey>
               <kpiName>?</kpiName>
               <kpiReport>?</kpiReport>
               <rising>?</rising>
               <interval>?</interval>
               <compoundOperation>?</compoundOperation>
               <baselineEnabled>?</baselineEnabled>
               <baselineOperation>?</baselineOperation>
               <baselineWindowSize>?</baselineWindowSize>
               <scope>?</scope>
               <description>?</description>
               <enabled>?</enabled>
```

```
<continuousAlarmEnabled>?</continuousAlarmEnabled>
<alarmType>?</alarmType>
<probableCause>?</probableCause>
<alarmNature>?</alarmNature>
<alarmScript>?</alarmScript>
<mailFrom>?</mailFrom>
<mailTo>?</mailTo>
<mailSubject>?</mailSubject>
<msgText>?</msgText>
<Monday>?</Monday>
<Tuesday>?</Tuesday>
<Wednesday>?</Wednesday>
<Thursday>?</Thursday>
<Friday>?</Friday>
<Saturday>?</Saturday>
<Sunday>?</Sunday>
<beginTime>?</beginTime>
<endTime>?</endTime>
<criticalOnset>?</criticalOnset>
<criticalOnsetOccurs>?</criticalOnsetOccurs>
<criticalAbate>?</criticalAbate>
<criticalAbateOccurs>?</criticalAbateOccurs>
<criticalPolicyOverride>?</criticalPolicyOverride>
<majorOnset>?</majorOnset>
<majorOnsetOccurs>?</majorOnsetOccurs>
<majorAbate>?</majorAbate>
<majorAbateOccurs>?</majorAbateOccurs>
<majorPolicyOverride>?</majorPolicyOverride>
<minorOnset>?</minorOnset>
<minorOnsetOccurs>?</minorOnsetOccurs>
<minorAbate>?</minorAbate>
<minorAbateOccurs>?</minorAbateOccurs>
<minorPolicyOverride>?</minorPolicyOverride>
<warningOnset>?</warningOnset>
<warningOnsetOccurs>?</warningOnsetOccurs>
<warningAbate>?</warningAbate>
<warningAbateOccurs>?</warningAbateOccurs>
<warningPolicyOverride>?</warningPolicyOverride>
<informationalOnset>?</informationalOnset>
<informationalOnsetOccurs>?</informationalOnsetOccurs>
<informationalAbate>?</informationalAbate>
<informationalAbateOccurs>?</informationalAbateOccurs>
<informationalPolicyOverride>?</informationalPolicyOverride>
<kpis>
   <!--1 or more repetitions:-->
   <KPIInfo>
      <columnName>?</columnName>
      <reportKey>?</reportKey>
      <kpiName>?</kpiName>
      <kpiReport>?</kpiReport>
      <rising>?</rising>
      <baselineEnabled>?</baselineEnabled>
      <baselineOperation>?</baselineOperation>
      <baselineWindowSize>?</baselineWindowSize>
      <scope>?</scope>
      <!--Optional:-->
      <criticalOnset>?</criticalOnset>
      <!--Optional:-->
      <criticalOnsetOperator>?</criticalOnsetOperator>
      <!--Zero or more repetitions:-->
      <criticalOnsetTestValue>?</criticalOnsetTestValue>
      <!--Optional:-->
      <criticalAbate>?</criticalAbate>
      <!--Optional:-->
```

```
                        <criticalAbateOperator>?</criticalAbateOperator>
                        <!--Zero or more repetitions:-->
                        <criticalAbateTestValue>?</criticalAbateTestValue>
                        <!--Optional:-->
                        <majorOnset>?</majorOnset>
                        <!--Optional:-->
                        <majorOnsetOperator>?</majorOnsetOperator>
                        <!--Zero or more repetitions:-->
                        <majorOnsetTestValue>?</majorOnsetTestValue>
                        <!--Optional:-->
                        <majorAbate>?</majorAbate>
                        <!--Optional:-->
                        <majorAbateOperator>?</majorAbateOperator>
                        <!--Zero or more repetitions:-->
                        <majorAbateTestValue>?</majorAbateTestValue>
                        <!--Optional:-->
                        <minorOnset>?</minorOnset>
                        <!--Optional:-->
                        <minorOnsetOperator>?</minorOnsetOperator>
                        <!--Zero or more repetitions:-->
                        <minorOnsetTestValue>?</minorOnsetTestValue>
                        <!--Optional:-->
                        <minorAbate>?</minorAbate>
                        <!--Optional:-->
                        <minorAbateOperator>?</minorAbateOperator>
                        <!--Zero or more repetitions:-->
                        <minorAbateTestValue>?</minorAbateTestValue>
                        <!--Optional:-->
                        <warningOnset>?</warningOnset>
                        <!--Optional:-->
                        <warningOnsetOperator>?</warningOnsetOperator>
                        <!--Zero or more repetitions:-->
                        <warningOnsetTestValue>?</warningOnsetTestValue>
                        <!--Optional:-->
                        <warningAbate>?</warningAbate>
                        <!--Optional:-->
                        <warningAbateOperator>?</warningAbateOperator>
                        <!--Zero or more repetitions:-->
                        <warningAbateTestValue>?</warningAbateTestValue>
                        <!--Optional:-->
                        <informationalOnset>?</informationalOnset>
                        <!--Optional:-->
                        <informationalOnsetOperator>?</informationalOnsetOperator>
                        <!--Zero or more repetitions:-->
                        <informationalOnsetTestValue>?</informationalOnsetTestValue>
                        <!--Optional:-->
                        <informationalAbate>?</informationalAbate>
                        <!--Optional:-->
                        <informationalAbateOperator>?</informationalAbateOperator>
                        <!--Zero or more repetitions:-->
                        <informationalAbateTestValue>?</informationalAbateTestValue>
                     </KPIInfo>
                  </kpis>
                  <tenants>
                     <!--1 or more repetitions:-->
                     <Tenant>?</Tenant>
                  </tenants>
                  <excludedScopeKeys>?</excludedScopeKeys>
               </ThresholdInfo>
            </thresholdInfoList>
         </ppm:editThresholdInfoList>
      </soapenv:Body>
   </soapenv:Envelope>
```

# enableThreshold

An example of the NBAPI enableThreshold method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:enableThreshold>
         <userThresholdName>snmp_avail_test</userThresholdName>
         <interval>15MIN</interval>
         <scope>Node=10.75.169.12</scope>
      </ppm:enableThreshold>
   </soapenv:Body>
</soapenv:Envelope>
```

# disableThreshold

An example of the NBAPI disableThreshold method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:disableThreshold>
         <userThresholdName>snmp_avail_test</userThresholdName>
         <interval>15MIN</interval>
         <scope>Node=10.75.169.12</scope>
      </ppm:disableThreshold>
   </soapenv:Body>
</soapenv:Envelope>
```

# rearmThreshold

An example of the NBAPI rearmThreshold method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:rearmThreshold>
         <userThresholdName>snmp_avail_test</userThresholdName>
         <interval>15MIN</interval>
         <scope>Node=10.75.169.12</scope>
      </ppm:rearmThreshold>
   </soapenv:Body>
</soapenv:Envelope>
```

# deleteThreshold

An example of the NBAPI deleteThreshold method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:deleteThreshold>
         <userThresholdName>snmp_avail_test</userThresholdName>
```

```
            <interval>15MIN</interval>
            <scope>Node=10.75.169.12</scope>
         </ppm:deleteThreshold>
      </soapenv:Body>
   </soapenv:Envelope>
```

# getThresholdInfo

An example of the NBAPI getThresholdInfo method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getThresholdInfo>
         <userThresholdName>IPSLAHTTP_ResponseTime</userThresholdName>
         <interval>15MIN</interval>
         <scope>Default</scope>
      </ppm:getThresholdInfo>
   </soapenv:Body>
</soapenv:Envelope>
```

# getAllThresholdInfo

An example of the NBAPI getAllThresholdInfo method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getAllThresholdInfo/>
   </soapenv:Body>
</soapenv:Envelope>
```

# getFilteredThresholdInfo

An example of the NBAPI getFilteredThresholdInfo method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getFilteredThresholdInfo>
         <userThresholdName>?</userThresholdName>
         <columnName>?</columnName>
         <reportKey>?</reportKey>
         <rising>?</rising>
         <interval>?</interval>
         <baselineEnabled>?</baselineEnabled>
         <baselineOperation>?</baselineOperation>
         <baselineWindowSize>?</baselineWindowSize>
         <scope>?</scope>
         <description>?</description>
         <enabled>?</enabled>
         <continuousAlarmEnabled>?</continuousAlarmEnabled>
         <alarmType>?</alarmType>
         <probableCause>?</probableCause>
         <alarmNature>?</alarmNature>
```

```
            <alarmScript>?</alarmScript>
            <mailFrom>?</mailFrom>
            <mailTo>?</mailTo>
            <mailSubject>?</mailSubject>
            <msgText>?</msgText>
         </ppm:getFilteredThresholdInfo>
      </soapenv:Body>
   </soapenv:Envelope>
```

# Group Management Methods

NBAPI methods that you can use to manage groups are provided in the following top8ics:



## addGroup

Example of the NBAPI addGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addGroup>
         <group>
            <name>?</name>
            <enabled>?</enabled>
            <processingSectionList>
               <!--1 or more repetitions:-->
               <GroupProcessingSection>
                  <name>?</name>
                  <type>?</type>
                  <matchingAlgorithm>?</matchingAlgorithm>
                  <matchingObjectList>
                     <!--1 or more repetitions:-->
                     <matchingObject>?</matchingObject>
                  </matchingObjectList>
                  <dataSourceList>
                     <!--1 or more repetitions:-->
                     <dateSource>?</dateSource>
                  </dataSourceList>
               </GroupProcessingSection>
            </processingSectionList>
         </group>
      </ppm:addGroup>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addGroup>
         <group>
```

```
            <name>group-2</name>
            <enabled>true</enabled>
            <processingSectionList>
               <!--1 or more repetitions:-->
               <GroupProcessingSection>
                  <name>section-1</name>
                  <type>type-1</type>
                  <matchingAlgorithm>If(Contains(cmStatusName, "online"), true,
false)</matchingAlgorithm>
                  <matchingObjectList>
                     <!--1 or more repetitions:-->
                     <matchingObject>ppm-vm1234</matchingObject>
                     <matchingObject>ppm-vm5678</matchingObject>
                  </matchingObjectList>
                  <dataSourceList>
                     <!--1 or more repetitions:-->
                     <dateSource>AGG_APN_CONTEXT_DATA_STATS</dateSource>
                     <dateSource>AGG_APN_CONTEXT_STATS</dateSource>
                  </dataSourceList>
               </GroupProcessingSection>
            </processingSectionList>
         </group>
      </ppm:addGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

# updateGroup

Example of the NBAPI updateGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updateGroup>
         <group>
            <name>?</name>
            <enabled>?</enabled>
            <processingSectionList>
               <!--1 or more repetitions:-->
               <GroupProcessingSection>
                  <name>?</name>
                  <type>?</type>
                  <matchingAlgorithm>?</matchingAlgorithm>
                  <matchingObjectList>
                     <!--1 or more repetitions:-->
                     <matchingObject>?</matchingObject>
                  </matchingObjectList>
                  <dataSourceList>
                     <!--1 or more repetitions:-->
                     <dateSource>?</dateSource>
                  </dataSourceList>
               </GroupProcessingSection>
            </processingSectionList>
         </group>
      </ppm:updateGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

## deleteGroup

Example of the NBAPI deleteGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:deleteGroup>
         <name>?</name>
      </ppm:deleteGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

## getGroupInfo

Example of the NBAPI getGroupInfo method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getGroupInfo>
         <groupNameList>
            <!--1 or more repetitions:-->
            <name>?</name>
         </groupNameList>
      </ppm:getGroupInfo>
   </soapenv:Body>
</soapenv:Envelope>
```

# Probe Management Methods

NBAPI methods that you can use to manage probes are listed in the following topics:

- addProbe, page 6-24
- updateProbe, page 6-26
- deleteProbe, page 6-26
- getProbeInfo, page 6-27

## addProbe

Example of the NBAPI addProbe method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addProbe>
         <probe>
            <PropertyList name="?">
               <!--1 or more repetitions:-->
               <Property name="?">
                  <!--You have a CHOICE of the next 2 items at this level-->
                  <Value>?</Value>
```

```
                  <ValueList/>
               </Property>
            </PropertyList>
         </probe>
      </ppm:addProbe>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addProbe>
         <probe>
               <PropertyList>
                  <Property name="Type">
                     <Value>HTTPProbe</Value>
                  </Property>
                  <Property name="Device">
                     <Value>ppm-lnx-vm012.cisco.com</Value>
                  </Property>
                  <Property name="Name">
                     <Value>HTTP-Probe-2</Value>
                  </Property>
                  <Property name="Description">
                     <Value>HTTP Probe 1</Value>
                  </Property>
                  <Property name="Enabled">
                     <Value>true</Value>
                  </Property>
                  <Property name="Interval">
                     <Value>15</Value>
                  </Property>
                  <Property name="OpenTimeout">
                     <Value>60</Value>
                  </Property>
                  <Property name="ResponseTimeout">
                     <Value>60</Value>
                  </Property>
                  <Property name="IPAddress">
                     <Value>google.com</Value>
                  </Property>
                  <Property name="Port">
                     <Value>80</Value>
                  </Property>
                  <Property name="Username">
                     <Value>username1</Value>
                  </Property>
                  <Property name="Password">
                     <Value>password2</Value>
                  </Property>
                  <Property name="HttpHeaderFieldTable">
                     <ValueList/>
                  </Property>
                   <Property name="HttpRequestMethod">
                     <Value>GET</Value>
                  </Property>
                   <Property name="HttpRequestVersion">
                     <Value>GET</Value>
                  </Property>
                  <Property name="HttpStatusCodeTable">
                     <ValueList>
                        <Property name="HttpStatusCodeRow">
                           <ValueList>
```

```
                                    <Property name="HttpStatusCode">
                                       <ValueList>
                                          <Property name="HttpStatusCodeRangeBegin">
                                             <Value>200</Value>
                                          </Property>
                                          <Property name="HttpStatusCodeRangeEnd">
                                             <Value>200</Value>
                                          </Property>
                                       </ValueList>
                                    </Property>
                                 </ValueList>
                              </Property>
                           </ValueList>
                        </Property>
                        <Property name="Protocol">
                           <Value>HTTPS</Value>
                        </Property>
                        <Property name="Application">
                           <Value>mail</Value>
                        </Property>
                        <Property name="UrlPath">
                           <Value>HTTPS://google.com:80/mail</Value>
                        </Property>
                     </PropertyList>
                  </probe>
               </ppm:addProbe>
            </soapenv:Body>
         </soapenv:Envelope>
```

# updateProbe

Example of the NBAPI updateProbe method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updateProbe>
         <probe>
            <PropertyList name="?">
               <!--1 or more repetitions:-->
               <Property name="?">
                  <!--You have a CHOICE of the next 2 items at this level-->
                  <Value>?</Value>
                  <ValueList/>
               </Property>
            </PropertyList>
         </probe>
      </ppm:updateProbe>
   </soapenv:Body>
</soapenv:Envelope>
```

# deleteProbe

Example of the NBAPI deleteProbe method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
```

```
      <ppm:deleteProbe>
         <probeId>
            <deviceName>?</deviceName>
            <probeName>?</probeName>
         </probeId>
      </ppm:deleteProbe>
   </soapenv:Body>
</soapenv:Envelope>
```

## getProbeInfo

An example of the NBAPI getProbeInfo method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getProbeInfo>
         <probeIdList>
            <!--1 or more repetitions:-->
            <probeId>
               <deviceName>?</deviceName>
               <probeName>?</probeName>
            </probeId>
         </probeIdList>
      </ppm:getProbeInfo>
   </soapenv:Body>
</soapenv:Envelope>
```

# Polling Group Management Methods

NBAPI methods that you can use to manage polling groups are listed in the following topics:

- addPollingGroup, page 6-27
- updatePollingGroup, page 6-28
- deletePollingGroup, page 6-28
- getPollingGroupInfo, page 6-28
- replacePalDeviceCapabilities, page 6-30
- getPalDeviceCapabilities, page 6-30

## addPollingGroup

An example of the NBAPI addPollingGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:ppm="http://cisco.com/ppm">

   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addPollingGroup>
         <PollingGroup>
            <name>polling-group-2</name>
            <timeout>60</timeout>
            <retries>2</retries>
```

```
            <snmpenabled>false</snmpenabled>
         </PollingGroup>
      </ppm:addPollingGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

# updatePollingGroup

An example of the NBAPI updatePollingGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:ppm="http://cisco.com/ppm">

   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updatePollingGroup>
         <PollingGroup>
            <name>polling-group-1</name>
            <timeout>45</timeout>
            <retries>1</retries>
            <snmpenabled>true</snmpenabled>
         </PollingGroup>
      </ppm:updatePollingGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

# deletePollingGroup

An example of the NBAPI deletePollingGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:ppm="http://cisco.com/ppm">

   <soapenv:Header/>
   <soapenv:Body>
      <ppm:deletePollingGroup>
         <PollingGroupId>polling-group-2</PollingGroupId>
      </ppm:deletePollingGroup>
   </soapenv:Body>
</soapenv:Envelope>
```

# getPollingGroupInfo

An example of the NBAPI getPollingGroup method is provided below:

```
<soapenv:Envelope xmlns:soapenv="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:ppm="http://cisco.com/ppm">

   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getPollingGroupInfo>
         <PollingGroupIdList>
            <!--1 or more repetitions:-->
            <name>polling-group-1</name>
            <name>polling-group-2</name>
         </PollingGroupIdList>
      </ppm:getPollingGroupInfo>
   </soapenv:Body>
```

```
</soapenv:Envelope>

<S:Envelope xmlns:S="
http://schemas.xmlsoap.org/soap/envelope/">

   <S:Body>
      <ns2:getPollingGroupInfoResponse xmlns:ns2="
http://cisco.com/ppm">

         <PollingGroupInfoList>
            <PollingGroup>
               <name>polling-group-1</name>
               <timeout>45</timeout>
               <retries>1</retries>
               <snmpenabled>false</snmpenabled>
            </PollingGroup>
            <PollingGroup>
               <name>polling-group-2</name>
               <timeout>60</timeout>
               <retries>2</retries>
               <snmpenabled>false</snmpenabled>
            </PollingGroup>
         </PollingGroupInfoList>
      </ns2:getPollingGroupInfoResponse>
   </S:Body>
</S:Envelope>

<soapenv:Envelope xmlns:soapenv="
http://schemas.xmlsoap.org/soap/envelope/" xmlns:ppm="http://cisco.com/ppm">

   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getPollingGroupInfo/>
   </soapenv:Body>
</soapenv:Envelope>

<S:Envelope xmlns:S="
http://schemas.xmlsoap.org/soap/envelope/">

   <S:Body>
      <ns2:getPollingGroupInfoResponse xmlns:ns2="
http://cisco.com/ppm">

         <PollingGroupInfoList>
            <PollingGroup>
               <name>Default</name>
               <timeout>30</timeout>
               <retries>2</retries>
               <snmpenabled>true</snmpenabled>
            </PollingGroup>
            <PollingGroup>
               <name>Linux</name>
               <timeout>30</timeout>
               <retries>2</retries>
               <snmpenabled>true</snmpenabled>
            </PollingGroup>
            <PollingGroup>
               <name>polling-group-1</name>
               <timeout>45</timeout>
               <retries>1</retries>
               <snmpenabled>false</snmpenabled>
            </PollingGroup>
            <PollingGroup>
               <name>polling-group-2</name>
```

```
               <timeout>60</timeout>
               <retries>2</retries>
               <snmpenabled>false</snmpenabled>
            </PollingGroup>
         </PollingGroupInfoList>
      </ns2:getPollingGroupInfoResponse>
   </S:Body>
</S:Envelope>
```

# replacePalDeviceCapabilities

An example of the replacePalDeviceCapabilities is provided below. It replaces the whole Protocol Abstraction Layer (PAL) file and also returns the original contents, so you can restore them later if needed.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:replacePalDeviceCapabilities>
         <palDeviceCapabilities>
            <key name="?">
               <!--You have a CHOICE of the next 2 items at this level-->
              <!--Zero or more repetitions - key elements can contain key elements and/or
entry elements in any order:-->
               <key/>
               <!--Zero or more repetitions:-->
               <entry name="?">?</entry>
            </key>
         </palDeviceCapabilities>
      </ppm:replacePalDeviceCapabilities>
   </soapenv:Body>
</soapenv:Envelope>

Response:

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:replacePalDeviceCapabilitiesResponse xmlns:ns2="http://cisco.com/ppm">
         <palDeviceCapabilities>
... Same content as from getPalDeviceCapabilities below ...
         </palDeviceCapabilities>
      </ns2:replacePalDeviceCapabilitiesResponse>
   </S:Body>
</S:Envelope>
```

# getPalDeviceCapabilities

An example of the NBAPI getPalDeviceCapabilities method used to retrieve a device PAL information. The method has no parameters and always returns the whole file.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getPalDeviceCapabilities/>
   </soapenv:Body>
</soapenv:Envelope>
Response:
```

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getPalDeviceCapabilitiesResponse xmlns:ns2="http://cisco.com/ppm">
         <palDeviceCapabilities>
            <key name="device-capability">
               <key name="default-properties">
                  <entry name="software">IOS</entry>
                  <entry name="DEVICE_TIMEOUT">60000</entry>
                  <entry name="CLI_CONNECTION_TIME">60</entry>
                  <entry name="CLI_SSH1_ENCRYPTION_CIPHER">DES3</entry>
                  <entry name="CLI_TERMINAL_EMULATION">vt100</entry>
                  <entry name="CLI_ENABLE_CONDITION">PASSWORD</entry>
                  <entry name="DB_LOGIN_USERNAME"/>
                  <entry name="DB_LOGIN_PASSWORD"/>
..............................
       </palDeviceCapabilities>
```

# Report Management

Report management methods are provided in the following topics:



## setReportStatus

An example of the NBAPI setReportStatus method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category></category>
         <status></status>
         <device></device>
         <intervals></intervals>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category>Availability: ICMP Ping</category>
         <status>enable</status>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category>Availability: ICMP Ping</category>
         <status>disable</status>
         <intervals>hourly daily</intervals>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category>Availability: ICMP Ping</category>
         <status>enable</status>
         <device>ppm-ucs-vm20</device>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category>Availability: ICMP Ping</category>
         <status>disable</status>
         <device>10.81.82.99</device>
         <intervals>15min hourly monthly</intervals>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category>all</category>
         <status>disable</status>
         <device>ppm-ucs-vm20</device>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:setReportStatus>
         <category>all</category>
         <status>disable</status>
      </ppm:setReportStatus>
   </soapenv:Body>
</soapenv:Envelope>
```

# assignReportPolicies

An example of the NBAPI assignReportPolicies method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:assignReportPolicies>
         <DeviceAssignmentList>
            <!--1 or more repetitions:-->
            <DeviceAssignment>
               <device>?</device>
               <assignment>?</assignment>
            </DeviceAssignment>
         </DeviceAssignmentList>
      </ppm:assignReportPolicies>
   </soapenv:Body>
</soapenv:Envelope>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:assignReportPoliciesResponse xmlns:ns2="http://cisco.com/ppm"/>
   </S:Body>
</S:Envelope>
```

# deleteReportPolicies

An example of the NBAPI deleteReportPolicies method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:deleteReportPolicies>
         <NameList>
            <!--1 or more repetitions:-->
            <name>?</name>
         </NameList>
      </ppm:deleteReportPolicies>
   </soapenv:Body>
</soapenv:Envelope>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:deleteReportPoliciesResponse xmlns:ns2="http://cisco.com/ppm"/>
   </S:Body>
</S:Envelope>
```

# getReportPolicies

An example of the NBAPI getReportPolicies method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getReportPolicies>
         <ReportPolicyFilter>
```

```
            <!--Optional:-->
            <PolicyName>?</PolicyName>
            <!--Optional:-->
            <DeviceName>?</DeviceName>
            <!--Optional:-->
            <CategoryName>?</CategoryName>
            <!--Optional:-->
            <ReportName>?</ReportName>
         </ReportPolicyFilter>
      </ppm:getReportPolicies>
   </soapenv:Body>
</soapenv:Envelope>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getReportPoliciesResponse xmlns:ns2="http://cisco.com/ppm">
         <ReportPolicyList>
            <DefaultReportPolicy>
               <MasterSetting>
                  <FlagSetting>
                     <Name>24_Hour_Time_Mode</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>master</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>1MIN</Name>
                     <Enabled>false</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>5MIN</Name>
                     <Enabled>false</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>15MIN</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>HOURLY</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>DAILY</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>WEEKLY</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>MONTHLY</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>CSV</Name>
                     <Enabled>false</Enabled>
                  </FlagSetting>
                  <FlagSetting>
                     <Name>DB</Name>
                     <Enabled>true</Enabled>
                  </FlagSetting>
                  <FlagSetting>
```

```
      <Name>Hourly_5_Minute_Reports</Name>
      <Enabled>false</Enabled>
   </FlagSetting>
   <FlagSetting>
      <Name>Hourly_15_Minute_Reports</Name>
      <Enabled>false</Enabled>
   </FlagSetting>
   <FlagSetting>
      <Name>Disk_Checking</Name>
      <Enabled>true</Enabled>
   </FlagSetting>
   <NumberSetting>
      <Name>STATS_AGING_1MIN</Name>
      <Value>2</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>EXPORT_AGING_1MIN</Name>
      <Value>2</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>STATS_AGING_5MIN</Name>
      <Value>4</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>EXPORT_AGING_5MIN</Name>
      <Value>3</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>STATS_AGING_15MIN</Name>
      <Value>7</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>EXPORT_AGING_15MIN</Name>
      <Value>2</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>STATS_AGING_HOURLY</Name>
      <Value>7</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>EXPORT_AGING_HOURLY</Name>
      <Value>14</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>STATS_AGING_DAILY</Name>
      <Value>94</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>EXPORT_AGING_DAILY</Name>
      <Value>94</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>STATS_AGING_WEEKLY</Name>
      <Value>730</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>EXPORT_AGING_WEEKLY</Name>
      <Value>730</Value>
   </NumberSetting>
   <NumberSetting>
      <Name>STATS_AGING_MONTHLY</Name>
      <Value>1825</Value>
   </NumberSetting>
   <NumberSetting>
```

```
            <Name>EXPORT_AGING_MONTHLY</Name>
            <Value>1825</Value>
         </NumberSetting>
         <NumberSetting>
            <Name>STATS_AGING_BULK</Name>
            <Value>14</Value>
         </NumberSetting>
         <NumberSetting>
            <Name>EXPORT_AGING_BULK</Name>
            <Value>14</Value>
         </NumberSetting>
      </MasterSetting>
      <CategorySetting>
         <CategoryName>Availability</CategoryName>
         <ReportSetting>
            <ReportName>HTTP Probe</ReportName>
            <FlagSetting>
               <Name>all</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>15MIN</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>HOURLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>DAILY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>WEEKLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>MONTHLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>DB</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
         </ReportSetting>
      </CategorySetting>
   </DefaultReportPolicy>
   <DeviceTypeReportPolicy>
      <Name>Cisco2821</Name>
      <CategorySetting>
         <CategoryName>Availability</CategoryName>
         <ReportSetting>
            <ReportName>HTTP Probe</ReportName>
            <FlagSetting>
               <Name>all</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>15MIN</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>HOURLY</Name>
               <Enabled>true</Enabled>
```

```
            </FlagSetting>
            <FlagSetting>
               <Name>DAILY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>WEEKLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>MONTHLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>DB</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
         </ReportSetting>
      </CategorySetting>
   </DeviceTypeReportPolicy>
   <CustomReportPolicy>
      <Name>test</Name>
      <CategorySetting>
         <CategoryName>Availability</CategoryName>
         <ReportSetting>
            <ReportName>HTTP Probe</ReportName>
            <FlagSetting>
               <Name>all</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>15MIN</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>HOURLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>DAILY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>WEEKLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>MONTHLY</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
            <FlagSetting>
               <Name>DB</Name>
               <Enabled>true</Enabled>
            </FlagSetting>
         </ReportSetting>
      </CategorySetting>
   </CustomReportPolicy>
   <SingleDeviceReportPolicy>
      <Name>ems-lnx212.cisco.com</Name>
      <CategorySetting>
         <CategoryName>Availability</CategoryName>
         <ReportSetting>
            <ReportName>HTTP Probe</ReportName>
            <FlagSetting>
```

```
                        <Name>all</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                     <FlagSetting>
                        <Name>15MIN</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                     <FlagSetting>
                        <Name>HOURLY</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                     <FlagSetting>
                        <Name>DAILY</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                     <FlagSetting>
                        <Name>WEEKLY</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                     <FlagSetting>
                        <Name>MONTHLY</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                     <FlagSetting>
                        <Name>DB</Name>
                        <Enabled>true</Enabled>
                     </FlagSetting>
                  </ReportSetting>
               </CategorySetting>
            </SingleDeviceReportPolicy>
         </ReportPolicyList>
      </ns2:getReportPoliciesResponse>
   </S:Body>
</S:Envelope>
```

# updateReportPolicies

The updateReportPolicies file should contain an xml ReportPolicyList produced by the getReportPolicies call. The ordering of categories, reports, flags, and numbers is not significant. The tools always generates output in the same order but handles any input order as long as the types are in the correct order.

For each update flag set, you can specify that all intervals are enabled or disabled, or you can specify individual interval settings. You cannot do both. Trying to do both will cause a validation failure.

The updateReportPolicies operation does not change unspecified settings, so you do not need to specify any settings that you do not want to change.

An example of the NBAPI updateReportPolicies method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updateReportPolicies>
         <ReportPolicyList>
            <!--Optional:-->
            <DefaultReportPolicy>
               <!--Optional:-->
               <MasterSetting>
                  <!--Zero or more repetitions:-->
                  <FlagSetting>
```

```
            <Name>?</Name>
            <Enabled>?</Enabled>
         </FlagSetting>
         <!--Zero or more repetitions:-->
         <NumberSetting>
            <Name>?</Name>
            <Value>?</Value>
         </NumberSetting>
      </MasterSetting>
      <!--Zero or more repetitions:-->
      <CategorySetting>
         <CategoryName>?</CategoryName>
         <!--Zero or more repetitions:-->
         <ReportSetting>
            <ReportName>?</ReportName>
            <!--Zero or more repetitions:-->
            <FlagSetting>
               <Name>?</Name>
               <Enabled>?</Enabled>
            </FlagSetting>
         </ReportSetting>
      </CategorySetting>
   </DefaultReportPolicy>
   <!--Zero or more repetitions:-->
   <DeviceTypeReportPolicy>
      <Name>?</Name>
      <!--1 or more repetitions:-->
      <CategorySetting>
         <CategoryName>?</CategoryName>
         <!--Zero or more repetitions:-->
         <ReportSetting>
            <ReportName>?</ReportName>
            <!--Zero or more repetitions:-->
            <FlagSetting>
               <Name>?</Name>
               <Enabled>?</Enabled>
            </FlagSetting>
         </ReportSetting>
      </CategorySetting>
   </DeviceTypeReportPolicy>
   <!--Zero or more repetitions:-->
   <CustomReportPolicy>
      <Name>?</Name>
      <!--1 or more repetitions:-->
      <CategorySetting>
         <CategoryName>?</CategoryName>
         <!--Zero or more repetitions:-->
         <ReportSetting>
            <ReportName>?</ReportName>
            <!--Zero or more repetitions:-->
            <FlagSetting>
               <Name>?</Name>
               <Enabled>?</Enabled>
            </FlagSetting>
         </ReportSetting>
      </CategorySetting>
   </CustomReportPolicy>
   <!--Zero or more repetitions:-->
   <SingleDeviceReportPolicy>
      <Name>?</Name>
      <!--1 or more repetitions:-->
      <CategorySetting>
         <CategoryName>?</CategoryName>
         <!--Zero or more repetitions:-->
```

```
                  <ReportSetting>
                     <ReportName>?</ReportName>
                     <!--Zero or more repetitions:-->
                     <FlagSetting>
                        <Name>?</Name>
                        <Enabled>?</Enabled>
                     </FlagSetting>
                  </ReportSetting>
               </CategorySetting>
            </SingleDeviceReportPolicy>
         </ReportPolicyList>
      </ppm:updateReportPolicies>
   </soapenv:Body>
</soapenv:Envelope>

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:updateReportPoliciesResponse xmlns:ns2="http://cisco.com/ppm"/>
   </S:Body>
</S:Envelope>
```

# User Management Methods

Prime Performance Manager north bound API (NBAPI) allows you to perform some user management functions including adding, updating, and deleting users and managing user parameters. The functionality is exactly the same as user management using the Prime Performance Manager GUI. For example, the allowed values for RoleName are the role display strings in the GUI. The main difference is that all NBAPI methods work in batch mode, so you can submit changes for multiple users in one request. Aside from getUsers, the methods all return an empty message unless an error occurs. getUsers returns a UserList, same as the one submitted for editUsers. Errors are returned if user access is not enabled, or if the requester does not have appropriate permission, which is same as required to use the GUI for user management.

URL: https://*ppm_gateway*:4440/nbapi/admin

**Note** If Prime Performance Manager is integrated into the Cisco Prime Carrier Management suite, the interface is https://*ppm_gateway*:4440/nbapi/event.

**Note** User access must be enabled. SOAP requests must include a Prime Performance Manager user name and password through the HTTP Basic Authorization Header. The user must have sufficient permissions to perform the requested SOAP action.

Project wsdl:

/opt/CSCOppm-gw/tomcat/webapps/nbapi/WEB-INF/wsdl/AdminAPI.wsdl

NBAPI user management method examples are provided in the following topics:

- updateUserPasswords, page 6-43

# addUsers

An example of the NBAPI addUsers method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:addUsers>
         <AddUsersRequest>
            <!--1 or more repetitions:-->
            <UserAddSpecification>
               <UserName>?</UserName>
               <RoleName>?</RoleName>
               <!--Optional:-->
               <Password>?</Password>
               <!--Optional:-->
               <ForceChangeAtNextLogin>?</ForceChangeAtNextLogin>
               <!--Optional:-->
               <ForceAdd>?</ForceAdd>
               <!--Optional:-->
               <UserDetails>
                  <!--Optional:-->
                  <PasswordAging>?</PasswordAging>
                  <!--Optional:-->
                  <FirstName>?</FirstName>
                  <!--Optional:-->
                  <LastName>?</LastName>
                  <!--Optional:-->
                  <Email>?</Email>
                  <!--Optional:-->
                  <Phone>?</Phone>
                  <!--Optional:-->
                  <CustomerName>?</CustomerName>
                  <!--Optional:-->
                  <AccountNumber>?</AccountNumber>
                  <!--Optional:-->
                  <GroupName>?</GroupName>
                  <!--Optional:-->
                  <HomeViewName>?</HomeViewName>
                  <!--Optional:-->
                  <Tenants>
                     <!--1 or more repetitions:-->
                     <Tenant>?</Tenant>
                  </Tenants>
               </UserDetails>
            </UserAddSpecification>
         </AddUsersRequest>
      </ppm:addUsers>
   </soapenv:Body>
</soapenv:Envelope>
```

# deleteUsers

An example of the NBAPI deleteUsers method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
```

```
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:deleteUsers>
         <UserNameList>
            <!--1 or more repetitions:-->
            <name>?</name>
         </UserNameList>
      </ppm:deleteUsers>
   </soapenv:Body>
</soapenv:Envelope>
```

# editUsers

An example of the NBAPI editUsers method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:editUsers>
         <UserList>
            <!--1 or more repetitions:-->
            <User>
               <UserName>?</UserName>
               <!--Optional:-->
               <RoleName>?</RoleName>
               <!--Optional:-->
               <UserDetails>
                  <!--Optional:-->
                  <PasswordAging>?</PasswordAging>
                  <!--Optional:-->
                  <FirstName>?</FirstName>
                  <!--Optional:-->
                  <LastName>?</LastName>
                  <!--Optional:-->
                  <Email>?</Email>
                  <!--Optional:-->
                  <Phone>?</Phone>
                  <!--Optional:-->
                  <CustomerName>?</CustomerName>
                  <!--Optional:-->
                  <AccountNumber>?</AccountNumber>
                  <!--Optional:-->
                  <GroupName>?</GroupName>
                  <!--Optional:-->
                  <HomeViewName>?</HomeViewName>
                  <!--Optional:-->
                  <Tenants>
                     <!--1 or more repetitions:-->
                     <Tenant>?</Tenant>
                  </Tenants>
               </UserDetails>
            </User>
         </UserList>
      </ppm:editUsers>
   </soapenv:Body>
</soapenv:Envelope>
```

# getUsers

An example of the NBAPI getUsers method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:getUsers>
         <UserFilter>
            <!--Optional:-->
            <UserName>?</UserName>
         </UserFilter>
      </ppm:getUsers>
   </soapenv:Body>
</soapenv:Envelope>
```

# updateUserFlags

An example of the NBAPI updateUserFlags method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updateUserFlags>
         <UpdateUserFlagsRequest>
            <FlagName>?</FlagName>
            <UserNameList>
               <!--1 or more repetitions:-->
               <name>?</name>
            </UserNameList>
         </UpdateUserFlagsRequest>
      </ppm:updateUserFlags>
   </soapenv:Body>
</soapenv:Envelope>
```

# updateUserPasswords

An example of the NBAPI updateUserPasswords method is provided below:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ppm="http://cisco.com/ppm">
   <soapenv:Header/>
   <soapenv:Body>
      <ppm:updateUserPasswords>
         <UpdateUserPasswordsRequest>
            <!--1 or more repetitions:-->
            <UserPasswordSpecification>
               <UserName>?</UserName>
               <Password>?</Password>
               <!--Optional:-->
               <ForceChangeAtNextLogin>?</ForceChangeAtNextLogin>
            </UserPasswordSpecification>
         </UpdateUserPasswordsRequest>
      </ppm:updateUserPasswords>
   </soapenv:Body>
</soapenv:Envelope>
```

# View Management Methods

Use the following methods to add, delete, edit, and get views using the Prime Performance Manager north bound API (NBAPI).



## addView

An example of the NBAPI addView method is provided below:

```
<ppm:addView>
         <View name="?" interval="Min5Min15HourlyDailyWeeklyMonthly"
collectionInterval="15" reportId="?" age="?" createSchema="false" distribAgg="false"
context="Network,Node" category="?" sortWeight="8122001" reportType="none" textProps="?"
sourceFileName="?" numberOfColumns="2" goLiveDisabled="false" invisible="false"
showCompact="false" networkCacheEnabled="true" groupByFilterEnabled="false"
dashboardId="?" DisplayName="?" Owner="?" EditableGroups="?" VisibleGroups="?"
ExternalKey="?" Visible="false" Editable="false" WorkingView="false" OrigFile="?"
LegendEnabled="?">
            <!--Optional:-->
            <Criteria name="records">?</Criteria>
            <!--Zero or more repetitions:-->
            <Filter name="records" item="?" eq="?">?</Filter>
            <TableView baseTable="?">
               <!--You have a CHOICE of the next 13 items at this level-->
               <Column name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Column>
               <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
               <Bytes name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bytes>
               <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
              <IPFlags name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</IPFlags>
```

```
               <IpAddr name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</IpAddr>
               <Label name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Label>
              <IdLabel name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</IdLabel>
               <Time name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Time>
               <Key name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false">?</Key>
               <MetaColumn name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</MetaColumn>
               <Link name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false" linkReport="?">?</Link>
               <HeaderRow name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</HeaderRow>
            </TableView>
            <!--Optional:-->
           <GraphView graphsPerRow="2" hideMax="?" hideAvg="?" hideMin="?" showStdDev="?"
showTotal="?" showVar="?" showCurrent="?" graphsToMerge="-1">
               <!--Optional:-->
               <GraphSummary title="?" minimized="?" showLegend="false" seriesLimit="?"/>
               <!--You have a CHOICE of the next 1 items at this level-->
               <Graph title="?" matchSelector="?" yAxisKey="?" baseTable="?" hideMax="?"
hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?" showCurrent="?"
type="line">
                  <!--You have a CHOICE of the next 5 items at this level-->
```

```
                <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
                <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
                <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                <!--0 to 5 repetitions:-->
                <ReferenceLine value="?" name="?" color="?" opacity="?" rgline="false"/>
            </Graph>
            <!--You have a CHOICE of the next 1 items at this level-->
            <LeafGraph title="?" matchSelector="?" yAxisKey="?" baseTable="?"
hideMax="?" hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?"
showCurrent="?" type="line" showLegend="false">
                <!--You have a CHOICE of the next 5 items at this level-->
                <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
                <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
```

```
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
                  <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                  <!--0 to 5 repetitions:-->
                  <ReferenceLine value="?" name="?" color="?" opacity="?" rgline="false"/>
               </LeafGraph>
            </GraphView>
            <!--Optional:-->
            <FQDN>?</FQDN>
            <!--Optional:-->
            <FQDNId>-1</FQDNId>
            <!--1 to 9 repetitions:-->
            <WebReport name="?" interval="Min5Min15HourlyDailyWeeklyMonthly"
collectionInterval="15" reportId="?" age="?" createSchema="false" distribAgg="false"
context="Network,Node" category="?" sortWeight="8122001" reportType="none" textProps="?"
sourceFileName="?" numberOfColumns="2" goLiveDisabled="false" invisible="false"
showCompact="false" networkCacheEnabled="true" groupByFilterEnabled="false">
               <!--Optional:-->
               <Criteria name="records">?</Criteria>
               <!--Zero or more repetitions:-->
               <Filter name="records" item="?" eq="?">?</Filter>
               <TableView baseTable="?">
                  <!--You have a CHOICE of the next 13 items at this level-->
                  <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                  <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
                  <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                  <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
                  <IPFlags name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IPFlags>
```

```
                  <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                  <Label name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Label>
                  <IdLabel name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IdLabel>
                 <Time name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Time>
                  <Key name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false">?</Key>
                  <MetaColumn name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</MetaColumn>
                 <Link name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false" linkReport="?">?</Link>
                  <HeaderRow name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</HeaderRow>
               </TableView>
               <!--Optional:-->
               <GraphView graphsPerRow="2" hideMax="?" hideAvg="?" hideMin="?"
showStdDev="?" showTotal="?" showVar="?" showCurrent="?" graphsToMerge="-1">
                  <!--Optional:-->
                  <GraphSummary title="?" minimized="?" showLegend="false"
seriesLimit="?"/>
                  <!--You have a CHOICE of the next 1 items at this level-->
```

```
                <Graph title="?" matchSelector="?" yAxisKey="?" baseTable="?"
hideMax="?" hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?"
showCurrent="?" type="line">
                    <!--You have a CHOICE of the next 5 items at this level-->
                    <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                    <Bits name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bits>
                    <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                    <Util name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Util>
                    <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                    <!--0 to 5 repetitions:-->
                    <ReferenceLine value="?" name="?" color="?" opacity="?"
rgline="false"/>
                </Graph>
                <!--You have a CHOICE of the next 1 items at this level-->
                <LeafGraph title="?" matchSelector="?" yAxisKey="?" baseTable="?"
hideMax="?" hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?"
showCurrent="?" type="line" showLegend="false">
                    <!--You have a CHOICE of the next 5 items at this level-->
                    <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                    <Bits name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bits>
```

```
                    <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                    <Util name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Util>
                    <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                    <!--0 to 5 repetitions:-->
                    <ReferenceLine value="?" name="?" color="?" opacity="?"
rgline="false"/>
                </LeafGraph>
             </GraphView>
             <!--Optional:-->
             <FQDN>?</FQDN>
             <!--Optional:-->
             <FQDNId>-1</FQDNId>
          </WebReport>
          <!--1 or more repetitions:-->
          <Report>
             <!--1 or more repetitions:-->
             <SubReference/>
             <ReportName>?</ReportName>
             <GraphTitle>?</GraphTitle>
             <Aggregates>
                <ShowTotal95Percent>false</ShowTotal95Percent>
                <ShowSeries95Percent>false</ShowSeries95Percent>
                <ShowAverage>false</ShowAverage>
                <ShowTotal>false</ShowTotal>
                <ShowSLA>false</ShowSLA>
                <ShowForecast>false</ShowForecast>
             </Aggregates>
             <Legends>
                <showMax>false</showMax>
                <showAvg>false</showAvg>
                <showMin>false</showMin>
                <showStdDev>false</showStdDev>
                <showTotal>false</showTotal>
                <showVar>false</showVar>
                <showDefault>false</showDefault>
                <showCurrent>false</showCurrent>
             </Legends>
             <Forecast>
                <useCustomDate>false</useCustomDate>
                <customDay>0</customDay>
                <customHour>0</customHour>
                <customMinute>0</customMinute>
                <ampm>AM</ampm>
                <diffunit>Hours</diffunit>
                <diffval>1</diffval>
             </Forecast>
```

```
                <!--Zero or more repetitions:-->
                <Axis>
                   <enable>?</enable>
                   <byDefault>true</byDefault>
                   <!--Optional:-->
                   <min>?</min>
                   <!--Optional:-->
                   <max>?</max>
                   <!--Zero or more repetitions:-->
                   <seriesIndexes>?</seriesIndexes>
                </Axis>
                <!--Optional:-->
                <FQDN>?</FQDN>
                <!--Optional:-->
                <SeriesIDs>?</SeriesIDs>
                <Interval>?</Interval>
                <Duration>?</Duration>
                <TopStatSel>?</TopStatSel>
                <OutputType>?</OutputType>
                <StartDate>?</StartDate>
                <EndDate>?</EndDate>
                <RepeatType>?</RepeatType>
                <ReferenceID>?</ReferenceID>
                <DisplayType>CHART</DisplayType>
                <ShadeType>?</ShadeType>
                <CustomName>?</CustomName>
                <CustomSubtitle>?</CustomSubtitle>
                <CustomInterval>?</CustomInterval>
                <Filter>?</Filter>
                <MultiAxis>?</MultiAxis>
                <ShowTCAs>?</ShowTCAs>
                <!--0 to 5 repetitions:-->
                <ReferenceLine value="?" name="?" color="?" opacity="?" rgline="false"/>
                <ShowVerticalGrid>false</ShowVerticalGrid>
                <ShowTimeBar>false</ShowTimeBar>
                <ShowLogAxis>false</ShowLogAxis>
             </Report>
             <!--1 or more repetitions:-->
             <View/>
             <!--1 or more repetitions:-->
             <ActiveList>?</ActiveList>
          </View>
       </ppm:addView>
```

# deleteView

An example of the NBAPI deleteView method is provided below:

```
<ppm:deleteView>
<viewName>?</viewName>
</ppm:deleteView>
```

# editView

An example of the NBAPI editView method is provided below:

```
<ppm:editView>
         <View name="?" interval="Min5Min15HourlyDailyWeeklyMonthly"
collectionInterval="15" reportId="?" age="?" createSchema="false" distribAgg="false"
context="Network,Node" category="?" sortWeight="8122001" reportType="none" textProps="?"
```

```
sourceFileName="?" numberOfColumns="2" goLiveDisabled="false" invisible="false"
showCompact="false" networkCacheEnabled="true" groupByFilterEnabled="false"
dashboardId="?" DisplayName="?" Owner="?" EditableGroups="?" VisibleGroups="?"
ExternalKey="?" Visible="false" Editable="false" WorkingView="false" OrigFile="?"
LegendEnabled="?">
            <!--Optional:-->
            <Criteria name="records">?</Criteria>
            <!--Zero or more repetitions:-->
            <Filter name="records" item="?" eq="?">?</Filter>
            <TableView baseTable="?">
               <!--You have a CHOICE of the next 13 items at this level-->
               <Column name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Column>
               <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
               <Bytes name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bytes>
               <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
              <IPFlags name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</IPFlags>
               <IpAddr name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</IpAddr>
               <Label name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Label>
              <IdLabel name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</IdLabel>
               <Time name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
```

```
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Time>
               <Key name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false">?</Key>
               <MetaColumn name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</MetaColumn>
               <Link name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false" linkReport="?">?</Link>
               <HeaderRow name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</HeaderRow>
            </TableView>
            <!--Optional:-->
           <GraphView graphsPerRow="2" hideMax="?" hideAvg="?" hideMin="?" showStdDev="?"
showTotal="?" showVar="?" showCurrent="?" graphsToMerge="-1">
               <!--Optional:-->
               <GraphSummary title="?" minimized="?" showLegend="false" seriesLimit="?"/>
               <!--You have a CHOICE of the next 1 items at this level-->
               <Graph title="?" matchSelector="?" yAxisKey="?" baseTable="?" hideMax="?"
hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?" showCurrent="?"
type="line">
                  <!--You have a CHOICE of the next 5 items at this level-->
                  <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                 <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
                  <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
```

```
                <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
                <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                <!--0 to 5 repetitions:-->
                <ReferenceLine value="?" name="?" color="?" opacity="?" rgline="false"/>
              </Graph>
              <!--You have a CHOICE of the next 1 items at this level-->
              <LeafGraph title="?" matchSelector="?" yAxisKey="?" baseTable="?"
hideMax="?" hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?"
showCurrent="?" type="line" showLegend="false">
                <!--You have a CHOICE of the next 5 items at this level-->
                <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
                <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
                <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                <!--0 to 5 repetitions:-->
                <ReferenceLine value="?" name="?" color="?" opacity="?" rgline="false"/>
              </LeafGraph>
            </GraphView>
            <!--Optional:-->
            <FQDN>?</FQDN>
            <!--Optional:-->
            <FQDNId>-1</FQDNId>
            <!--1 to 9 repetitions:-->
```

```
            <WebReport name="?" interval="Min5Min15HourlyDailyWeeklyMonthly"
collectionInterval="15" reportId="?" age="?" createSchema="false" distribAgg="false"
context="Network,Node" category="?" sortWeight="8122001" reportType="none" textProps="?"
sourceFileName="?" numberOfColumns="2" goLiveDisabled="false" invisible="false"
showCompact="false" networkCacheEnabled="true" groupByFilterEnabled="false">
               <!--Optional:-->
               <Criteria name="records">?</Criteria>
               <!--Zero or more repetitions:-->
               <Filter name="records" item="?" eq="?">?</Filter>
               <TableView baseTable="?">
                  <!--You have a CHOICE of the next 13 items at this level-->
                  <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                  <Bits name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Bits>
                  <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                  <Util name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Util>
                  <IPFlags name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IPFlags>
                  <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                  <Label name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Label>
                  <IdLabel name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
```

```
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IdLabel>
                  <Time name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?">?</Time>
                  <Key name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false">?</Key>
                  <MetaColumn name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</MetaColumn>
                  <Link name="?" compoundname="?" id="?" sortable="true" descending="true"
tcaRising="true" colSpan="1" default="false" defaultValue="?" groupQueryOperation="?"
groupQueryOrder="desc" thresholdable="?" colValueNumeric="true" filterable="?"
nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0" nocolor="false"
downColors="false" floor="false" round="false" showPercentageColumn="false"
isReference="false" key="false" hideSeries="false" runningValue="?" context="?"
hideSummary="false" hideLegend="false" linkReport="?">?</Link>
                  <HeaderRow name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</HeaderRow>
               </TableView>
               <!--Optional:-->
               <GraphView graphsPerRow="2" hideMax="?" hideAvg="?" hideMin="?"
showStdDev="?" showTotal="?" showVar="?" showCurrent="?" graphsToMerge="-1">
                  <!--Optional:-->
                  <GraphSummary title="?" minimized="?" showLegend="false"
seriesLimit="?"/>
                  <!--You have a CHOICE of the next 1 items at this level-->
                  <Graph title="?" matchSelector="?" yAxisKey="?" baseTable="?"
hideMax="?" hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?"
showCurrent="?" type="line">
                     <!--You have a CHOICE of the next 5 items at this level-->
                     <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                     <Bits name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bits>
```

```
                    <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                    <Util name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Util>
                    <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                    <!--0 to 5 repetitions:-->
                    <ReferenceLine value="?" name="?" color="?" opacity="?"
rgline="false"/>
                </Graph>
                <!--You have a CHOICE of the next 1 items at this level-->
                <LeafGraph title="?" matchSelector="?" yAxisKey="?" baseTable="?"
hideMax="?" hideAvg="?" hideMin="?" showStdDev="?" showVar="?" showTotal="?"
showCurrent="?" type="line" showLegend="false">
                    <!--You have a CHOICE of the next 5 items at this level-->
                    <Column name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Column>
                    <Bits name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bits>
                    <Bytes name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Bytes>
                    <Util name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</Util>
                    <IpAddr name="?" compoundname="?" id="?" sortable="true"
descending="true" tcaRising="true" colSpan="1" default="false" defaultValue="?"
groupQueryOperation="?" groupQueryOrder="desc" thresholdable="?" colValueNumeric="true"
filterable="?" nodatavalue="?" colNameGraphPrefix="false" decimalPrecision="0"
```

```
nocolor="false" downColors="false" floor="false" round="false"
showPercentageColumn="false" isReference="false" key="false" hideSeries="false"
runningValue="?">?</IpAddr>
                    <!--0 to 5 repetitions:-->
                    <ReferenceLine value="?" name="?" color="?" opacity="?"
rgline="false"/>
                </LeafGraph>
             </GraphView>
             <!--Optional:-->
             <FQDN>?</FQDN>
             <!--Optional:-->
             <FQDNId>-1</FQDNId>
          </WebReport>
          <!--1 or more repetitions:-->
          <Report>
             <!--1 or more repetitions:-->
             <SubReference/>
             <ReportName>?</ReportName>
             <GraphTitle>?</GraphTitle>
             <Aggregates>
                <ShowTotal95Percent>false</ShowTotal95Percent>
                <ShowSeries95Percent>false</ShowSeries95Percent>
                <ShowAverage>false</ShowAverage>
                <ShowTotal>false</ShowTotal>
                <ShowSLA>false</ShowSLA>
                <ShowForecast>false</ShowForecast>
             </Aggregates>
             <Legends>
                <showMax>false</showMax>
                <showAvg>false</showAvg>
                <showMin>false</showMin>
                <showStdDev>false</showStdDev>
                <showTotal>false</showTotal>
                <showVar>false</showVar>
                <showDefault>false</showDefault>
                <showCurrent>false</showCurrent>
             </Legends>
             <Forecast>
                <useCustomDate>false</useCustomDate>
                <customDay>0</customDay>
                <customHour>0</customHour>
                <customMinute>0</customMinute>
                <ampm>AM</ampm>
                <diffunit>Hours</diffunit>
                <diffval>1</diffval>
             </Forecast>
             <!--Zero or more repetitions:-->
             <Axis>
                <enable>?</enable>
                <byDefault>true</byDefault>
                <!--Optional:-->
                <min>?</min>
                <!--Optional:-->
                <max>?</max>
                <!--Zero or more repetitions:-->
                <seriesIndexes>?</seriesIndexes>
             </Axis>
             <!--Optional:-->
             <FQDN>?</FQDN>
             <!--Optional:-->
             <SeriesIDs>?</SeriesIDs>
             <Interval>?</Interval>
             <Duration>?</Duration>
             <TopStatSel>?</TopStatSel>
```

```
                  <OutputType>?</OutputType>
                  <StartDate>?</StartDate>
                  <EndDate>?</EndDate>
                  <RepeatType>?</RepeatType>
                  <ReferenceID>?</ReferenceID>
                  <DisplayType>CHART</DisplayType>
                  <ShadeType>?</ShadeType>
                  <CustomName>?</CustomName>
                  <CustomSubtitle>?</CustomSubtitle>
                  <CustomInterval>?</CustomInterval>
                  <Filter>?</Filter>
                  <MultiAxis>?</MultiAxis>
                  <ShowTCAs>?</ShowTCAs>
                  <!--0 to 5 repetitions:-->
                  <ReferenceLine value="?" name="?" color="?" opacity="?" rgline="false"/>
                  <ShowVerticalGrid>false</ShowVerticalGrid>
                  <ShowTimeBar>false</ShowTimeBar>
                  <ShowLogAxis>false</ShowLogAxis>
               </Report>
               <!--1 or more repetitions:-->
               <View/>
               <!--1 or more repetitions:-->
               <ActiveList>?</ActiveList>
            </View>
         </ppm:editView>
```

# getView

An example of the NBAPI getView method is provided below:

```
Request:

      <ppm:getView>
         <viewName>?</viewName>
      </ppm:getView>

Response:

      <ns2:getViewResponse xmlns:ns2="http://cisco.com/ppm">
         <View DisplayName="Sample View" Editable="true" Owner="dalmccal" Visible="true"
name="dalmccal_._dalmccal_._Sample View" sourceFileName="etc/views/sampleView.xml"
textProps="sampleView">
            <FQDN>Node=</FQDN>
            <FQDNId>-1</FQDNId>
            <View DisplayName="Sub View One" Editable="true" Owner="dalmccal"
Visible="true" name="dalmccal_._dalmccal_._Sample View_._Sub View One"
sourceFileName="etc/views/sampleView.xml" textProps="sampleView">
               <FQDN>Node=SampleFQDN</FQDN>
               <FQDNId>-1</FQDNId>
               <Report>
                  <ReportName>level1Applications,level2SNMP,wrnSNMPPktsRate</ReportName>
                  <GraphTitle>gtSNMPPktsSec</GraphTitle>
                  <FQDN>Node=ems7609f.cisco.com,SystemName=ems7609f.cisco.com</FQDN>
                  <Interval>QUARTER_HOUR</Interval>
                  <Duration>last3Days</Duration>
                  <MultiAxis>false</MultiAxis>
                  <ShowTCAs>false</ShowTCAs>
                  <ShowVerticalGrid>false</ShowVerticalGrid>
                  <ShowTimeBar>false</ShowTimeBar>
                  <ShowLogAxis>false</ShowLogAxis>
```

```
                  <tenantId>0</tenantId>
               </Report>
               <Report>
                  <ReportName>level1Resources,level2CPU,wrnCPUUtil</ReportName>
                  <GraphTitle>gtCPUUtilPeak</GraphTitle>
                  <Interval>HOUR</Interval>
                  <Duration>last7Days</Duration>
                  <MultiAxis>false</MultiAxis>
                  <ShowTCAs>false</ShowTCAs>
                  <ShowVerticalGrid>false</ShowVerticalGrid>
                  <ShowTimeBar>false</ShowTimeBar>
                  <ShowLogAxis>false</ShowLogAxis>
                  <tenantId>0</tenantId>
               </Report>
              <View DisplayName="Sub View One Level One" Editable="true" Owner="dalmccal"
Visible="true" name="dalmccal_._dalmccal_._Sample View_._Sub View One_._Sub View One Level
One" sourceFileName="etc/views/sampleView.xml" textProps="sampleView">
                  <FQDN>Node=</FQDN>
                  <FQDNId>-1</FQDNId>
               </View>
            </View>
            <View DisplayName="Sub View Two" Editable="true" Owner="dalmccal"
Visible="true" name="dalmccal_._dalmccal_._Sample View_._Sub View Two"
sourceFileName="etc/views/sampleView.xml" textProps="sampleView">
               <FQDN>Node=</FQDN>
               <FQDNId>-1</FQDNId>
            </View>
            <View DisplayName="Sub View Three" Editable="true" Owner="dalmccal"
Visible="true" name="dalmccal_._dalmccal_._Sample View_._Sub View Three"
sourceFileName="etc/views/sample36View.xml" textProps="sampleView">
               <FQDN>Node=ASecondSampleFQDN</FQDN>
               <FQDNId>-1</FQDNId>
            </View>
         </View>
      </ns2:getViewResponse>
```


# Supporting Cisco Data Collections Manager

The following topics explain how to configure Cisco Data Collection Manager in Prime Performance Manager:



## Overview to DCM

The Cisco Data Collection Manager (DCM) provides a data collection framework for collecting data from Cisco devices. The DCM supports a profile-based configuration that allows you to set parameters for collecting, processing, and exporting the data.

The following notes and examples will help you implement DCM processing in Prime Performance Manager.

- Only IOS 15.3(1)T/PI20 (and IOS-XE 38 - 15.3(1)S - 3.8.0S support DCM 2.0
- Prime Performance Manager only supports DCM CPU and Memory reports.
- Use the DcmPoll macro to parse DCM files. See the cpu.xml and memoryPool.xml files for examples.
- Prime Performance Manager integrates DCM by parsing the DCM file with BulkStats.
- In SystemCapability.xml, the DCM capability indicate if the device supports DCM 2.0:

```
DCM = xmlPollPersist("DcmProfileStats", "ppm-profile");
```

  - DcmProfileStats is the PAL capability poll key defined in etc/palRuntime/conf/DeviceCapability.xml.
  - ppm-profile is the DCM profile name configured in the device and defined in properties/BulkStats.properties with the key name DCM_PROFILE_NAME

  Because DCM is only available in Cisco devices, define DCM capabilities in CiscoDevices section. Other capabilities related to DCM should utilize this definition, for example:

```
DCM_CPU = DCM
@and xmlPollPersist("DcmDataGroupStats", "CPU-cpmCPUTotalTable")
@and xmlPollPersist("DcmDataGroupStats", "CPU-cpmCPUThresholdTable");
```

  - DcmDataGroupStats is the PAL capability poll key defined in DeviceCapability.xml.

  - CPU-cpmCPUTotalTable and CPU-cpmCPUThresholdTable are the data-group names configured in the device. CPU refers to the poller name defined in cpu.xml. cpmCPUTotalTable refers to the cache name of the DcmPoll return data as below,

```
cpmCPUTotalTable = dcmPoll("cpmCPUTotalIndex,
cpmCPUTotalPhysicalIndex,
cpmCPUTotal5minRev,
cpmCPUTotal1minRev");
```

    Therefore, data-group name = poller name + '-' + cache name. If the data-group length name is longer than 32 characters, only the first 32 are valid.

- For information about the DCM profile structure the configure command format, see the *Cisco Data Collection Manager Configuration Guide*.
- DCM configuration notes.
  - All DCM entity names must be less than 32 characters.
  - The profile name must be sthe ame with the value of DCM_PROFILE_NAME in properties/BulkStats.properties.
  - Data groups cannot share one instance for known DCM issues.
  - The hostname configured in the device must not contain periods "." even though periods are valid for hostnames.
  - DCM aging can be modified from the Prime Performance Manager GUI on the Reports/Group Settings tab Bulk Stats Aging parameters.

# DCM Parameters

Table 7-1 shows the DCM parameters in properties and bulkstats.properties files.

***Table 7-1 DCM Parameters***

| Property | Description |
|---|---|
| DCM_DROP_DIR | The directory where DCM pushes the performance management file. |
| DCM_PROFILE_NAME | The DCM profile prefix name configured in the device.<br>Prime Performance Manager uses the prefix to filter DCM files. |
| DCM_DIR_MONITOR_REFRESH_INTERVAL | The rate (in seconds) at which the DCM Bulk Stats directory is monitored for new files. The default is 15 seconds. |
| DCM_DATAGROUP_INTERVAL | The data-group polling interval in seconds. This defines the interval at which of DCM polls device performance management data. The default is 60 seconds. |
| DCM_PROFILE_FILE_SIZE | Maximum buffer-size in bytes. Generally, one file should contain all performance monitoring data in one device interval. The default is 10240000 bytes. |
| DCM_PROFILE_FILE_RETAIN | The number of time(s) the DCM file retained in device is used with the retry transfer file. The default is 300. |
| DCM_PROFILE_FILE_RETRY | Number of times to retry transfer in case of transfer failure to both the primary and secondary URLs. Retries will take effect only if the retention of file is configured using retain. The default is 5 |

*Table 7-1 DCM Parameters*

| Property | Description |
|---|---|
| DCM_PROFILE_FILE_URL_PRIMARY | The primary URL where the DCM file is pushed. DCM only pushes the file to the primary URL if it works. The default is ftp://root:password@ipaddress |
| DCM_PROFILE_FILE_URL_SECONDARY | The secondary URL where the DCM file is pushed. DCM only pushes the file to the secondary URL if the primrary does not work. The default is ftp://root:password@ipaddress |
| DCM_PROFILE_INTERVAL_PROCESS | Sets the time period, in seconds, at which process files are created for the processed data and queued for transfer. The default is 1800 seconds. |
| DCM_PROFILE_INTERVAL_RAW | The period in seconds when the active file is frozen and queued for transfer. The default is 60 seconds. |

If you only want to collect five-minute report data, modify the following parameters as follows:,

- DCM_DIR_MONITOR_REFRESH_INTERVAL = 100
- DCM_DATAGROUP_INTERVAL = 300
- DCM_PROFILE_INTERVAL_RAW = 300

In addition, make Prime Performance Manager must read/write privileges on DCM_DROP_DIR.

# DCM Configuration for CPU Reports

The following examples shows a DCM configuration for CPU reports using a 60-second interval.

```
bulkstat data CPU-cpmCPUTotalTable type snmp
 object 1.3.6.1.4.1.9.9.109.1.1.1.1.2
 object 1.3.6.1.4.1.9.9.109.1.1.1.1.8
 object 1.3.6.1.4.1.9.9.109.1.1.1.1.7
exit
bulkstat instance CPU-cpmCPUTotalTable type snmp
 wildcard
exit
bulkstat data-group CPU-cpmCPUTotalTable
 interval polling 60
 collect type snmp data CPU-cpmCPUTotalTable instance CPU-cpmCPUTotalTable
exit

bulkstat data CPUOLD-cpmCPUTotalTable type snmp
 object 1.3.6.1.4.1.9.9.109.1.1.1.1.2
 object 1.3.6.1.4.1.9.9.109.1.1.1.1.4
 object 1.3.6.1.4.1.9.9.109.1.1.1.1.5
exit
bulkstat instance CPUOLD-cpmCPUTotalTable type snmp
 wildcard
exit
bulkstat data-group CPUOLD-cpmCPUTotalTable
 interval polling 60
 collect type snmp data CPUOLD-cpmCPUTotalTable instance CPUOLD-cpmCPUTotalTable
exit

bulkstat data CPU-cpmCPUThresholdTable type snmp
 object 1.3.6.1.4.1.9.9.109.1.2.4.1.2
 object 1.3.6.1.4.1.9.9.109.1.2.4.1.4
```

```
exit
bulkstat instance CPU-cpmCPUThresholdTable type snmp
 wildcard
exit
bulkstat data-group CPU-cpmCPUThresholdTable
 interval polling 60
 collect type snmp data CPU-cpmCPUThresholdTable instance CPU-cpmCPUThresholdTable
exit

bulkstat data CPUOLD-cpmCPUThresholdTable type snmp
 object 1.3.6.1.4.1.9.9.109.1.2.4.1.2
 object 1.3.6.1.4.1.9.9.109.1.2.4.1.4
exit
bulkstat instance CPUOLD-cpmCPUThresholdTable type snmp
 wildcard
exit
bulkstat data-group CPUOLD-cpmCPUThresholdTable
 interval polling 60
 collect type snmp data CPUOLD-cpmCPUThresholdTable instance CPUOLD-cpmCPUThresholdTable
exit

bulkstat profile ppm-profile
 file size 10240000
 file retain memory 600
 file transfer url primary ftp://admin:admin@10.10.10.10
 file transfer url secondary ftp://root:root@10.10.10.20
 file transfer retry 5
 interval transfer raw 60
 data-group CPU-cpmCPUTotalTable
 data-group CPUOLD-cpmCPUTotalTable
 data-group CPU-cpmCPUThresholdTable
 data-group CPUOLD-cpmCPUThresholdTable
exit
```

# DCM Configuration for Memory Reports

The following shows a sample DCM configuration for memory reports using a 60-second interval::

```
bulkstat data EnhancedMemoryPool-ciscoEnhMemor type snmp
 object 1.3.6.1.4.1.9.9.221.1.1.1.1.7
 object 1.3.6.1.4.1.9.9.221.1.1.1.1.8
 object 1.3.6.1.4.1.9.9.221.1.1.1.1.9
exit
bulkstat instance EnhancedMemoryPool-ciscoEnhMemor type snmp
 wildcard
exit
bulkstat data-group EnhancedMemoryPool-ciscoEnhMemor
 interval polling 60
 collect type snmp data EnhancedMemoryPool-ciscoEnhMemor instance
EnhancedMemoryPool-ciscoEnhMemor
exit

bulkstat data UCD_SNMP_MEMORY-memTable type snmp
 object 1.3.6.1.4.1.2021.4.5
 object 1.3.6.1.4.1.2021.4.6
 object 1.3.6.1.4.1.2021.4.11
exit
bulkstat instance UCD_SNMP_MEMORY-memTable type snmp
 wildcard
exit
```

```
bulkstat data-group UCD_SNMP_MEMORY-memTable
 interval polling 60
 collect type snmp data UCD_SNMP_MEMORY-memTable instance UCD_SNMP_MEMORY-memTable
exit

bulkstat data MemoryBuffer-cempMemBufferPoolTa type snmp
 object 1.3.6.1.4.1.9.9.221.1.1.2.1.3
 object 1.3.6.1.4.1.9.9.221.1.1.2.1.21
 object 1.3.6.1.4.1.9.9.221.1.1.2.1.22
exit
bulkstat instance MemoryBuffer-cempMemBufferPoolTa type snmp
 wildcard
exit
bulkstat data-group MemoryBuffer-cempMemBufferPoolTa
 interval polling 60
 collect type snmp data MemoryBuffer-cempMemBufferPoolTa instance
MemoryBuffer-cempMemBufferPoolTa
exit

bulkstat data MemoryPool-ciscoMemoryPoolTable type snmp
 object 1.3.6.1.4.1.9.9.48.1.1.1.2
 object 1.3.6.1.4.1.9.9.48.1.1.1.5
 object 1.3.6.1.4.1.9.9.48.1.1.1.6
 object 1.3.6.1.4.1.9.9.48.1.1.1.7
exit
bulkstat instance MemoryPool-ciscoMemoryPoolTable type snmp
 wildcard
exit
bulkstat data-group MemoryPool-ciscoMemoryPoolTable
 interval polling 60
 collect type snmp data MemoryPool-ciscoMemoryPoolTable instance
MemoryPool-ciscoMemoryPoolTable
exit

bulkstat data SystemExtMemory-cseSysMemoryPool type snmp
 object 1.3.6.1.4.1.9.9.305.1.1.2
exit
bulkstat instance SystemExtMemory-cseSysMemoryPool type snmp
 wildcard
exit
bulkstat data-group SystemExtMemory-cseSysMemoryPool
 interval polling 60
 collect type snmp data SystemExtMemory-cseSysMemoryPool instance
SystemExtMemory-cseSysMemoryPool
exit

bulkstat profile ppm-profile
 file size 10240000
 file retain memory 600
 file transfer url primary ftp://admin:admin@10.10.10.10
 file transfer url secondary ftp://root:root@10.10.10.20
 file transfer retry 5
 interval transfer raw 60
 data-group EnhancedMemoryPool-ciscoEnhMemor
 data-group UCD_SNMP_MEMORY-memTable
 data-group MemoryBuffer-cempMemBufferPoolTa
 data-group MemoryPool-ciscoMemoryPoolTable
 data-group SystemExtMemory-cseSysMemoryPool
exit
```


# XML Reference

This chapter provides information about the XML elements and tags used in Prime Performance Manager reports. Topics include:



## XML Elements

The following topics describe the XML elements used in Prime Performance Manager reports:

- WebReport, page A-5

# AggregatedDBSummary

The AggregatedDBSummary tag is used for the Aggregating the values available in the ProcessDBSummary. These are mainly used for the group reports.

# Bytes

The Bytes tag can be used to display the values in B/KB/MB/GB format in the Table and Graph view. They are mainly used for the size related parameters.

# Column

The Column element is contained in a CSV, Graph view, and Table view section and specifies the heading and table column for a column shown in the CSV report.

# Criteria

The Criteria element specifies the MIB capabilities used in the report, as defined in the *SystemCapability.xml* file and the *UserCapability.xml* file.

You can use the Criteria element within the Poll element and within the WebReport element.

# CSV

The CSV element contains elements and attributes that specify the format and content of the CSV report that is saved to a file when a user chooses the CSV report format.

The CSV element contains the following elements:

- Column
- Bits
- Bytes
- Util
- IpAddr
- Time

# Filter

The Filter tag can be used inside the WebReports to filter out a particular key value and display that specific reports alone.

# DecimalPrecision

Sets the decimal precision in Graph and Table views. For example, decimalPrecision =3 will set the decimal value as 3, for example, 1500.231 and 12.369. The level of decimal precision displayed in Prime Performance Manager reports is the greater value specified by the decimalPrecision element or by the user in User Preferences.

Graph and Table view section examples:

<GraphView>

```
       <GraphSummary title="vpnErrorPercentage" />
       <Graph title="vpnAvgSendErrorPercentage">
          <Util name="sendErrorPercentage" decimalPrecision = "3">txErrorPercent</Util>
       </Graph>
       <Graph title="vpnAvgRecvErrorPercentage">
          <Util name="recvErrorPercentage"  decimalPrecision = "3">rxErrorPercent</Util>
       </Graph>
       <LeafGraph title="vpnAvgSendRecvErrorPercentage">
          <Util name="vpnAvgSendErrorPercentage" decimalPrecision =
"3">txErrorPercent</Util>
          <Util name="vpnAvgRecvErrorPercentage"  decimalPrecision =
"3">rxErrorPercent</Util>
       </LeafGraph>
    </GraphView>

    <TableView baseTable="L3VPN">
      <IdLabel/>
      <Label colSpan="1" name=""/>
      <Label colSpan="3" name="send"/>
      <Label colSpan="3" name="receive"/>
      <HeaderRow/>
      <Link name="node"      context="Node">fqdnid</Link>
      <Link name="vrfName"   context="vrfName">vrfName</Link>
      <Link name="interface" context="l3vpnIfDescr">l3vpnIfDescr</Link>
      <Time/>
      <Column                name="type">ifType.mibEnum()</Column>
      <Column                name="goodPackets"                  decimalPrecision =
"3">txTotPkts</Column>
      <Column               name="errors"                              decimalPrecision
= "3">txErrors</Column>
      <Util   default="true" name="errorPercentage" decimalPrecision =
"3">txErrorPercent</Util>
      <Column                name="goodPackets"                  decimalPrecision =
"3">rxTotPkts</Column>
      <Column               name="errors"                              decimalPrecision
= "3">rxErrors</Column>
      <Util                        name="errorPercentage"            decimalPrecision =
"3">rxErrorPercent</Util>
    </TableView>
```

# graphsPerRow

The graphsPerRow attribute is defined inside the Graph view tag for the number of graphs to be displayed in the row. These are mainly used in the Dashboard reports.

## graphsToMerge

The graphsToMerge attribute is defined inside the GraphView tag for the number of graphs to merge and display. It is mainly used for Dashboard reports.

## GraphView

Specifies the graphs and subgraphs used in the graph view for a report. Contains the following elements:

- Column
- Graph
- GraphSummary
- LeafGraph

## IpAddr

Specifies an IP address. Used in the Column element.

## IdLabel

Used in the Column element.

## LeafGraph

Specifies the attributes of a LeafGraph, which is a graph that shows multiple values. Used in the GraphView element.

## Link

Specifies a link. Used in the TableView section.

## PollDefinition

The PollDefinition section specifies what MIB variables are polled for the report, and can call reports macros used to filter polling.

## PollerList

The PollerList element is the main element in the report.xml file. It encapsulates all of the other elements.

# Poll

The Poll element specifies the following key report elements:

- Criteria
- PollDefinition
- ProcessPollResult
- ProcessDBSummary

# ProcessDBSummary

The ProcessDBSummary contains code that builds the virtual table used to hold report data in memory. At the end of each defined reporting period, the table data is written to the physical database on the gateway.

# ProcessPollResult

The ProcessPollResult section contains code that specifies how MIB data is processed.

# TableView

The TableView element specifies the contents and text used in the Table report view. The TableView element contains the following elements:

- Column
- Label
- Link
- Util
- Byte

# Util

Specifies a utility that calculates a value, such as an average. Used in the GraphView or TableView element.

# WebReport

The WebReport element specifies the elements and text strings used in a Graph view of the report as well as the Table view. The WebReport section contains GraphView and TableView.

# XML Tags

Table A-1 shows the XML tags used in Prime Performance Manager reports. Most tags specify the default behavior within a report. These values override the preferences set in the Prime Performance Manager User Preferences window unless the user enables the Override Report Definitions option. For information about user preferences, see "Setting User Preferences" in the *Cisco Prime Performance Manager 1.7 User Guide*.

***Table A-1 XML Tags***

| Tag | Description |
|---|---|
| colNameGraphPrefix | Is contained in a TableView section. If colNameGraphPrefix=true, column values are added to graph titles such as Slot 1 CPU 2. If set to false (default), the graph title would be 1 | 2. This tag is useful for index integers that require descriptors to be useful in graph titles, for example, in IPSLA, CPU, memory, and other reports. Example:<br><br>`<TableView baseTable="IPSLA_JITTER">`<br>`<IdLabel/>`<br>`<Label colSpan="1" name="responseTime" />`<br>`<Label colSpan="1" name="jitter" />`<br>`<Label colSpan="1" name="packetLoss" />`<br>`<Label colSpan="1" name="reachability" />`<br><br>`<HeaderRow/>`<br>`<Link name="node" context="Node">fqdnid</Link>`<br>`<Link name="target" context="TargetAddress" colNameGraphPrefix="true">TargetAddress</Link>`<br>`<Link name="owner" context="IPSLAOwner" colNameGraphPrefix="true">IPSLAOwner</Link>`<br>`<Link name="tag" context="IPSLATag" colNameGraphPrefix="true">IPSLATag</Link>`<br>`<Link name="type" context="IPSLARTTType" hideSummary="true">IPSLARTTType</Link>`<br>`<Link name="index" context="IPSLAIndex" colNameGraphPrefix="true">IPSLAIndex</Link>`<br>`<Time/>`<br><br>`<Column default="true" name="millisecs">ResponseTimeAvg</Column>`<br>`<Column name="millisecs">(JitterPosSDAvg + JitterNegSDAvg + JitterPosDSAvg + JitterNegDSAvg) / 4</Column>`<br>`<Util name="percentage">PacketLossOverallUtilAvg</Util>`<br>`<Util name="percentage" tcaRising="false">TargetReachabilityUtilAvg</Util>`<br>`</TableView>` |

*Table A-1 XML Tags (continued)*

| Tag | Description |
|---|---|
| CsvPoll | Specifies that values should be read from a file not through SNMP. Example:<br>`ApnBearersStatsTable = CsvPoll("apn",`<br>`"integer:auth-req-sent,`<br>`integer:auth-acc-rcvd,`<br>`integer:auth-timeout,`<br>`integer:acc-req-sent,`<br>`integer:acc-rsp-rcvd,`<br>`integer:acc-req-timeout,`<br>`integer:act-defbear,`<br>`integer:act-dedbear,`<br>`integer:setup-defbear,`<br>`integer:setup-dedbear,`<br>`integer:rel-defbear,`<br>`integer:rel-dedbear,`<br>`integer:rel-fail-defbear,`<br>`integer:rel-fail-dedbear,`<br>`integer:rej-defbear,`<br>`integer:rej-dedbear,`<br>`integer:mod-uebear,`<br>`integer:mod-nwbear,`<br>`integer:ue-init-modfail,`<br>`integer:nw-init-modfail",`<br>`"string:vpnname,`<br>`integer:vpnid,`<br>`string:apn");` |
| descending | Works the same way as tcaRising by indicating which metric direction is good or bad. It is used to sort and color values in tables. If descending="false" values are sorted in ascending order instead of descending. descending=”true” is the default for tables. |
| downColors | Is applied to metrics such as down percentage. The math is similar to the descending/availability user preference. Here, the value checked determines whether the displayed color is the value minus one, not the value directly. For example, if an interface is down more than one percent of the time it needs to be identified. However, the descending metrics cannot be used because it measures the down time not up time.<br>This flag, if set to true, indicates colors should be displayed using the <value-1> formula. |
| hideAvg | If hideAvg=true, hides the average values in graph summary tables and legends, regardless of the user preference. If hideAvg=false, the user preference is followed. hideAvg goes on the Graph line. An example is shown in showTotal. |
| hideLegend | Can be added to report XML index columns to trim long, complex indexes in charts and graph subtitles, while displaying the index columns in summary tables. Some IPSLA and ClassMap reports are examples where not all MIB indexes need to be displayed because they rarely change or are index integers with little human value. Report components prefaced with hideLegend=true are not be displayed in graph legends. This tag applies to reports, dashboards, views, and StarGraphs. For an example, see hideSummary. |
| hideMax | If hideMax=true, hides the maximum values in graph summary tables and legends, regardless of the user preference. If hideMax=false, the user preference is followed. hideMax goes on the Graph line. An example is shown in showTotal. |
| hideMin | If hideMin=true, hides the minimum values in graph summary tables and legends, regardless of the user preference. If hideMin=false, the user preference is followed. hideMin goes on the Graph line. An example is shown in showTotal. |

*Table A-1 XML Tags (continued)*

| Tag | Description |
|---|---|
| hideSeries | hideSeries is used inside a LeafGraph so the LeafGraph appears with one data series as the primary display and the other data series is hidden. It is normally used with a KPI graph and contributing KPI counters. The KPI line is shown by default and the underlying counters are hidden. The user can click a counter data series and see which ones are contributing to the KPI and in which manner. hideSeries was designed for small cell KPI reports but can work for other KPI reports. It goes on the Column, Util or other lines inside a LeafGraph, not on the LeafGraph itself. Example:<br>`<LeafGraph title="gstrmsApcsRabSetupSuccess" showLegend="true">`<br>`<Util name="gencsRabSetupSuccess" descending="false" nocolor="true">`<br>`csRABSetupSuccessPercentage</Util>`<br>`<Column name="genrabSuccessCsVoice" hideSeries="true">rabSuccessCsVoice</Column>`<br>`<Column name="genrabSuccessCsVideo" hideSeries="true">rabSuccessCsVideo</Column>`<br>`<Column name="genrabAttemptsCsVoice" hideSeries="true">rabAttemptsCsVoice</Column>`<br>`<Column name="genrabAttemptsCsVideo" hideSeries="true">rabAttemptsCsVideo</Column>`<br>`</LeafGraph>` |
| hideSummary | Performs the same functions as hideLegend, but includes summary tables. hideSummary=true, hides the average values in summary tables and graph legends, regardless of the user preference. If hideSummary=false, the user preference is followed.<br>hideLegend and hideSummary appear in the Link line. Example:<br>`<Link name="index" context="IPSLAIndex" hideLegend="true" colNameGraphPrefix="true">IPSLAIndex</Link>`<br>`<Link name="type" context="IPSLARTTType" hideSummary="true">IPSLARTTType</Link>` |
| LastFlapTime_UNIXFormat | Displays the report time in Unix or POSIX time. GUI report example:<br>`<ProcessPollResult>`<br>`lastFlapTime_UNIXFormat = ccsCmFlapLastFlapTime.dateAndTime("UNIX_FORMAT");`<br>`</ProcessPollResult>`<br><br>`<ProcessDBSummary baseTableName="CMTS_CM_State_HOT">`<br>`<Var name="LastFlapTime_UNIXFormat" type="Long">lastFlapTime_UNIXFormat</Var>`<br>`</ProcessDBSummary>`<br><br>`<TableView baseTable="CMTS_CM_State_HOT">`<br>`<Time name="LastFlapTime_UNIXFormat" key="true">LastFlapTime_UNIXFormat</Time>`<br>`</TableView>`<br>CSV report example:<br>`<CSV name="CMTS_CM_State_HOT" location="gateway" listen="CMTS_CM_State_HOT">`<br>`<Time name="LastFlapTime_UNIXFormat" key="true">LastFlapTime_UNIXFormat</Time>`<br>`</CSV>` |

*Table A-1 XML Tags (continued)*

| Tag | Description |
|---|---|
| nocolor | Prime Performance Manager allows users to display utilization values in red, orange, and green, cooresponding to the utilization thresholds defined in User Preferences. This tag allows you to override that setting and always display the specified utilization value in black font and hide background colors in summary tables. If nocolor=true, ignore the user's red, green, and orange utlilization color preference and display the data in black font. Utilization xamples of values where this tag might be applied include:<br>• Error Percentage<br>• Discard Percentage<br>• Buffer Miss Percentage |
| ReferenceLine | Added to the Graph section, allows you to add a user-defined SLA or other custom reference line to charts. Format example:<br><Graph><br><ReferenceLine name="test" value="102" color="#FF0000" opacity="0.2"/><br></Graph><br>Color defaults to #000000; opacity defaults to 0.5. Name and value are required.<br>**Note** IE 8 does not support the opacity parameter. |
| showCurrent | If showCurrent=true, displays the current values in graph summary tables and legends, regardless of the user preference. If showCurrent=false, the user preference is followed. showCurrent goes on the Graph line. An example is shown in showTotal. |
| showLegend | If showLegend=true, displays legends in graphs, regardless of the user preference. If showLegend=false, the user preference is followed. showLegend goes on the LeafGraph line. An example is shown in hideSeries. |
| showMax | If showMax=true, displays the maximum values in graph summary tables and legends, regardless of the user preference. If showMax=false, the user preference is followed. showMax goes on the Graph line. An example is shown in showTotal. |
| showTotal | If showTotal=true, always displays total values in graph summary tables and legends, regardless of the user preference. If showTotal=false, the user preference is followed.<br>hideMin, hideMax, showMax, hideAvg, showCurrent, showMax, showTotal all go on the Graph line. Examples:<br>`<Graph title="gtCPUUtil1MinAvg" hideMin="false" showCurrent="true" showMax="true" hideAvg="false">`<br>`<Graph title="gtCPUUserUtil" hideMax="true">`<br>`<Graph title="gtCPUUtil1MinAvg" hideMin="false" showCurrent="true" showMax="true" hideAvg="false">` |
| tcaRising | For thresholdable fields, specifies whether the negative KPI value is rising or falling. For example, utilization going up and availability going down are negative. If tcaRising=true (default), the field KPI is rising. If tcaRising=false, the KPI is falling. You cannot use the tcaRising attribute for any LeafGraph fields. For more information on thresholdable fields, see Creating a Report with Thresholdable Fields, page 2-19. |

*Table A-1 XML Tags (continued)*

| Tag | Description |
|---|---|
| thresholdable | Specifies whether users can create thresholds for a data value. If thresholdable=true, users can create thresholds for it. If thresholdable=false, users cannot create thresholds.<br>tcaRising and thresholdable both go on a Value field inside either a GraphView or TableView section. Examples:<br>`<Util name="percentage" descending="false" tcaRising="false">TargetReachabilityUtilAvg</Util>`<br>`<Column name="voltage" thresholdable="false">ciscoEnvMonVoltageStatusValue</Column>` |
| XmlPoll | Tells the XML poller to push polling through SSH/Telnet. Example:<br>`bgpSummaryTable = XmlPoll("bgpStats", "bgpStats.bgpSummaryIos", "", "integer:tableVersion, integer:rpkiValid, integer:rpkiNotFound, integer:rpkiInvalid", "", false, "bgpIpv4SummaryTable");` |
| yAxisKey | Allows you to customize the Y-axis label name title for the graph report.<br>If yAxisKey="RPM", it will display Y-axis title as RPM for that graph report.<br>The **yAxixKey** title displays in the graph title, for example:<br>`<Graph title="gtUDPOutDatagrams" yAxisKey="RPM">` |


# Reports Macro Reference

The Prime Performance Manager report interface provides a number of predefined report macros that you can use in your reports. Macros can be called in two different ways:

1. object.macro (*arg1*, *arg2*, *arg3*, etc.) or
2. macro (*object*, *arg1*, *arg2*, *arg3*, etc.)

In the reference topics in this chapter, syntax for the second method is provided.

If arguments are enclosed in square brackets ([]), this indicates that the argument optional.

When it is stated that a certain argument is of a certain type (i.e, the object is a string type), that argument can also be replaced with:

- A macro/algorithm that returns the same type,

Or

- A numeric/string value if it is a numeric/string type

## Macros

# ADD

Syntax

**ADD** (object, arg1, arg2)

Macro Description

If object is an instance of a collection, the macro adds the arg1 data to object collection. If the object is an instance of Map, the macro adds the key-value (arg1-arg2) pair to object. Returns null, in case of a failure.

- object is either an object instance or a collection instance.
- arg1 is either a key or a value.
- arg2 is a value.

Example:

```
row.add("totalInPkts",totalInPkts);
```

# APPNAME

Syntax

APPNAME (arg1)

Macro Description

Queries the application name given the application ID.

- arg1 - is an application id.

**Example**:

appName = AppName(applicationId);

# APSTATSXMLPOLL

**Syntax**

APSTATSXMLPOLL (arg1, arg2, arg3)

Macro Description

Collects AP (Access Point) stats from 3GPP XML files.

- arg1: a string type, it's the filename of a properties file in etc/apstats/system/, example for RMS-LUS.properties, arg1 is "RMS-LUS", this properties file defines the parameters for the macro to pull AP stats files from RMS LUS server.
- arg2: a list of counters to be collected by this macro call, separated by comma.
- arg3: a list of key counters, separated by comma.

Example:

```
 apStatsXmlPoll("RMS-LUS","MAXCSPAGINGBURST,MAXPSPAGINGBURST,…,RABSUCCESSPSR99",
"apNodeId");
```

# ASNTONAME

**Syntax**

ASNTONAME (arg1)

Macro Description

Retrieves AS name (description) given an AS number.

- arg1 – is the AS number of Java Long type.

**Example:**

```
srcASName = ASNToName(srcAS);
```

# AVIPOLL

Syntax

AVIPoLL (arg1,arg2,arg3,arg4,arg5,arg6,arg7,arg8,arg9,arg10,arg11,arg12,arg13,arg14)

Macro Description

Used only in the PollDefinition section to retrieve AVI device report data. Parameters:

- param 0—Values that need to be polled
- param 1—REST API input parameters
- param 3—REST AP URL pattern for I
- param 4—The path to get results
- param 5—Static or dynamic configuration
- param 6—Map or list<map>
- param 7—REST API input metrics list
- param 8—REST API input parameters
- param 9—Additional REST API parameters
- param 10—Additional REST API parameters
- param 11—REST API header name
- param 12—REST API header name
- param 13—Cache key

**Example**:

```
seBandwidthTable  =   AviPoll("octetstring:header%entity_uuid,
octetstring:header%name,
octetstring:header%statistics%mean",
"header_entity_uuid",
"",
"/api/analytics/metrics/serviceengine/",
"series",
false,
false,
"metric_id=se_stats.avg_bandwidth",
seTable.getAll("uuid"),
"limit=1",
"step=300",
"X-Avi-Tenant",
seTable.getAll("results_name"),
"seBandwidthTable");
```

# BITS

Syntax

**BITS** (object, arg )

Macro Description

Returns true if the position in object identified by arg is 1; false, otherwise:

- object is bits string
- arg is an integer number of the bit position to match

**Example**:

```
status = cfmMdiMetricsValid.bits(0);
```

# BOTTOMN

Syntax

**BOTTOMN** (object, arg1, arg2)

Macro Description

Returns the bottom n (arg2) rows from object sorted by a sort key ascending:

- object is a table
- arg1 is what column to sort by
- arg2 is the number of records to get (n)..
- When used in the Filter section in a ProcessDBSummary section, rows is an implicit variable that typically can be used for an object that is set when ProcessDBSummary execution is complete.

**Example**:

```
rows = rows.bottomN("ProcessCPUUtil5mAvg", 5);
```

# BYTESTOMACADDR

**Syntax**

BYTESTOMACADDR (arg1)

Macro Description

Converts given address (in bytes) to MAC address (in String).

- arg1 – is the address in bytes.

**Example:**

```
srcMac  = BytesToMacAddr(sourceMacAddress);
```

# CHANGENODEIPADDRESSES

**Syntax**

CHANGENODEIPADDRESSES (ipAddrTable, ipAddressTable, cIpAddressTable.)

Macro Description

Used in conjunction with the report poller ipAddress.xml file. This macro updates the node IP addresses using the IP addresses polled from the corresponding report poller ipAddress.xml file where ipAddrTable, ipAddressTable and cIpAddressTable contain the results of polling and joining the interface and address tables from IF-MIB.my, IP-MIB.my, and CISCO-IETF-IP-MIB.my MIBs. Refer to the $SR/etc/pollers/system/ipAddress.xml poller configuration file for further information.

**Example:**

```
if(hasCapability("IP_ADDRESS_TABLE_IP_MIB") @or
   hasCapability("IP_ADDRESS_TABLE_CISCO_IETF_IP_MIB"),
   ChangeNodeIpAddresses(ipAddrTable, ipAddressTable, cIpAddressTable));
```

# CONTAINS

Syntax

**CONTAINS**(arg1,arg2)

**Macro Description**

Checks if arg1 contains arg2. If it is contained, it returns true, else it returns false.

- arg1 and arg2 may be Maps, Collections or Strings.

**Example**:

```
if(cmStatusName.contains("online"),true,false);
```

# COUNT

Syntax

**COUNT**(object)

**Macro Description**

Ipdates the rows of the grouped data and the count of these rows. Returns the count.

- object consists of the grouped data which needs to be counted.

Example:

```
cmStatusGroups.count();
```

# CREDSEXIST

**Syntax**

CREDSEXIST (arg1. arg2)

Macro Description

Checks whether specific credentials exist.

- arg1 – the protocol category
- arg2 – subsystem for credential

Example:

```
OPENSTACK_CAP = CredsExist("PALHttp","identity");
```

# CSVPULLNEXT

**Syntax**

CSVPULLNEXT (arg)

Macro Description

Pulls the device CSV, which is regularly updated with new lines. This macro parses the file and pulls new lines (after last pull) from it. The new lines are stored as a CSV file in the drop directory. The directory is monitored and parsed by the generic bulkstats feature. For example, in small cells, the PMG server inserts a new line to its pmg-perf-periodic.csv file every 20 seconds.

- arg1: a String type, it's the filename of a properties file in etc/csvPull/system/,

**Example:**

```
For pmg-perf.properties, the macro call is csvPullNext("pmg-perf");
The properties file pmg-perf.properties defines the CSV file location, filename etc.
parameters used by the macro to pull lines from the CSV file.
```

# DATEANDTIME

Syntax

**DATEANDTIME** ()

Macro Description

Returns the DateAndTime format defined in SNMPv2-TC or a time value in milliseconds defined in the Java Calendar.getTimeInMillis(). Detailed input information:

```
 Size list:1: 8
                2: 11
Display hint:2d-1d-1d,1d:1d:1d.1d,1a1d:1d
```

```
Description:A date-time specification.
        field  octets  contents                  range
        ----  ------  --------                    -----
         1      1-2   year                         0..65536
         2       3    month                       1..12
         3       4    day                            1..31
         4       5    hour                           0..23
         5       6    minutes                     0..59
         6       7    seconds                     0..60   (use 60 for leap-second)
         7       8    deci-seconds               0..9
         8       9    direction from UTC       '+' / '-'
         9      10    hours from UTC            0..11
        10      11    minutes from UTC        0..59
```

**Note** If only local time is known, time zone information (fields 8-10) is not present.

Example:

```
Tuesday May 26, 1992 at 1:30:15 PM EDT would be displayed as: 1992-5-26,13:30:15.0,-4:0
```

# DCMPOLL

**Syntax**

DCMPOLL (arg)

Macro Description

Parses the performance file in cache that is pushed from the DCM module in device.

- arg: is the set of snmp oid

**Example**:

```
cpmCPUTotalTable = dcmPoll("cpmCPUTotalIndex,
cpmCPUTotalPhysicalIndex,
pmCPUTotal5minRev,
pmCPUTotal1minRev");
```

# DELTA

Syntax

**DELTA** (*object*)

Macro Description

Returns the delta between the current and the previous poll value (currentPolling - previousPolling). Each time the poll is performed, the poll value is persisted for use as the 'previous' poll value the next time a poll is performed. The macro also takes care of conditions that can occur such as when the first poll occurs (no previous value) and when the number overflows.

- object parameter is a numeric type If detect wrap is enabled it will check for wraps and adjust before calculating difference.
- Used only in the ProcessPollResult section.

**Example:**

```
AuthenTransactionSuccesses     = casAuthenTransactionSuccesses.delta();
```

# DELTANEXT

Syntax

**DELTANEXT** (*object*, arg1)

Macro Description

Calls the DELTA macro between the next row in the database instead of the previous polling. Continues to calculate the delta until the row's value for arg1 changes.

- Object is a numeric type and arg1 is a string type, which is the name of an index of a previously created variable in the PollDefinition section (needs to be in the same table as *object*).

**Example:**

```
duration = rttMonIcmpJitterStatsStartTimeId..deltaNext("rttMonCtrlAdminIndex") / 100;
```

# DEVICECUSTOMNAME

Syntax

**DEVICECUSTOMNAME**()

Macro Description

Returns the custom name of the device.

**Example:**

```
deviceCustomName();
```

# DEVICEID

Syntax

**DEVICEID**()

Macro Description

Returns the unique ID of the device.

**Example:**

```
deviceID();
```

# DEVICEIPADDRESS

Syntax

**DEVICEIPADDRESS**()

Macro Description

Returns the IP address for the device in the current context.

**Example:**

The following entry determines if the device's IP addresses contains "102.168":

```
If(Contains(DeviceIpAddress(), "102.168"), true, false);
```

# DEVICELOCATION

Syntax

**DEVICELOCATION**()

Macro Description

Returns the device location attribute.

**Example:**

The following entry determines if the device's location attribute contains "labx":

```
If(Contains(DeviceLocation(), "labx"), true, false);
```

# DEVICENAME

Syntax

**DEVICENAME**()

Macro Description

Returns the device name.

Example:

The following entry determines if the device name attribute contains "ppm7000a":

```
If(Contains(DeviceName(), "ppm7000a"), true, false);
```

# DEVICESTATE

**Syntax**

DEVICESTATE (String arg)

Macro Description

Gets the Device State metrics including alarms and severities at the server level that is at the gateway. The macro may be invoked using any of the following options as parameter.

- arg maybe:
  - “Device” - provides severity counts for each device
  - “DeviceType” - provides severity counts for each device type
  - “Alarm Severity” - provides counts all alarm severities
  - “Highest Severity” - provides counts of only severities.

**Example:**

```
DeviceState(“Device”);
```

# DEVICESOFTWAREVERSION

Syntax

**DEVICESOFTWAREVERSION**()

Macro Description

Returns the device software version.

Example:

The following entry determines if the current device's software version contains 1.7.0:

```
If(Contains(DeviceSoftwareVersion(), "1.0.3"), true, false);
```

# DEVICESYNCNAME

Syntax

**DEVICESYNCNAME**()

Macro Description

Returns the device sync name.

Example:

The following entry determines if the device's sync name contains ppm7000a:

```
If(Contains(DeviceSyncName(), "ppm7000a"), true, false);
```

# DEVICESYSNAME

Syntax

**DEVICESYSNAME**()

Macro Description

Returns the device system name.

Example:

The following entry determines if the device's system name contains ppm7000a:

```
If(Contains(DeviceSysName(), "ppm7000a"), true, false);
```

# DEVICETYPE

Syntax

**DEVICETYPE** (object)

Macro Description

Returns the device type of the node, if found, else returns "unknown".

- object is the sysObjectID.
- Used in the SystemCapabilities.xml and UserCapability.xml files.

Example:

```
If((DeviceType() == "vmwESX"), Environment(" SNMPENV_MAX_VARBIND = 10;"), false);
```

# DIRNAMEPOLL

Syntax

DIRNAMEPOLL (arg1, arg2, arg3, arg4, arg5, arg6)

Macro Description

This class provides the directory name poll function.

- arg1 - package ID
- arg2 - action name
- arg3 - the name of the base directory
- arg4 - keys for values
- arg5 - static or dynamic config
- arg6 - cache key

**Example:**

```
hostNameTable           = dirNamePoll("collectdStats",
"collectdStats.ls",
"",
"dirName",
false,
"hostNameTable");
```

# DOUBLEVALUE

Syntax

**DOUBLEVALUE** (object)

Macro Description

Converts *object* into a double and returns it. Returns Null, if there is a failure.

Example:

```
satSigNoiseRatio  = saSatSigCndisp.doubleValue();
```

# ENDSWITH

Syntax

ENDSWITH (object, arg1)

Macro Description

Returns true if the string *object* ends with the string arg1. It is similar to Java's endsWith string function.

- object is a string type
- arg1 is a string.

Example:

```
memoryPoolInfo = memoryPoolInfo.filter(not(cempMemPoolName.endsWith("image")));
```

# ENTITYINFO

Syntax

**ENTITYINFO**()

Macro Description

Gets the data of ENTITY-MIB. If it is found in cache, the cache data will be returned. If not, the persistency data is checked. If found and the entity is not been changed since the last poll, the persisted data is returned. If still no data is available, the device is polled for the entity data.

Example:

```
entPhysicalTable = EntityInfo();
```

# EXPONENT

**Syntax**

**EXPONENT** (arg1, arg2)

Macro Description

Returns the exponent value. Returns Null, in case of a failure.

- arg1 is base and arg2 is exponent

Example:

```
exponentValue = Exponent("e", 2);
```

# FILTER

Syntax

**FILTER** (object, arg1)

Macro Description

Returns a subset of objects with items that do not pass the conditions (items that return false) removed.

- object is a table.
- arg1 is conditions (returns a Boolean result).

Example:

```
cgnStatisticsTable = cgnStatisticsTable.filter(true);
```

# FILTERCUSTOMINTERVAL

**Syntax**

FILTERCUSTOMINTERVAL (Object arg)

Macro Description

Filters data based upon a user specified filter.

- arg is the filter.

Example:

```
FilterCustomInterval("Sunday,Monday,Thursday;14:30").
The above will only pass data that is from 2:30 PM on Sunday, Monday, or Thursday.
```

# FLOWPOLL

Syntax

**FLOWPOLL** (arg1, arg2, arg3, arg4)

Macro Description

Used only in the PollDefinition section TO retrieve NetFlow statistics from receive NetFlow streams.

- arg1—A list of non-key fields (comma delimited list of field names in double quotes) to be retrieved from NetFlow streams.
- arg2—A list of key fields (comma delimited list of field names in double quotes) to be retrieved from NetFlow streams. A key field can optionally have a default value separated by a colon delimiter.
- arg3—A list of template IDs fields (comma delimited list of template numbers in double quotes). If key fields are not enough to identify unique flows by themselves, you can provide template IDs to act as additional filters. This argument is optional.
- arg4—A list of valid NetFlow versions (comma delimited list of version numbers in double quotes). If provided the macro returns rows only when the device's NetFlow version matches with one of the version numbers in this parameter list. This argument is optional.

Example:

```
flowPoll("octets,
pkts",
"input,
 srcaddr,
 flowDirection:0")

flowPoll("egressVRFID,
                    postNATSourceIPv4Address,
 postNAPTSourceTransportPort",
"ingressVRFID,
 sourceIPv4Address,
 sourceTransportPort,
 protocolIdentifier",
 "256,257");

flowPoll("octets,
pkts",
"input,
 srcaddr,
 flowDirection:0",
"",
"5,9");
```

# FOR

Syntax

**FOR** ( arg1,arg2, arg3, arg4)

Macro Description

The macro uses the arguments: loop initializer, loop control and loop increment, in a Java' for' loop and determines the return value of each processor. If the return value is a break, it breaks the Java for loop.

- arg1 is the loop initializer.
- arg2 is the loop control.
- arg3 is the loop increment.
- arg4 is the value, which indicates the processor.

Example:

```
for(index1=0,value @gt 0,index1 = index1 + 1, value = value / 10.0);.
```

# FOREACH

Syntax

**FOREACH** (object, arg1, arg2)

Macro Description

The macro uses a Java 'for' loop to iterate through each record of the object. Each object along with the arg1 is used to determine the return value. If the return value is a break, it breaks the Java 'for' loop.

- object is a list
- arg1 is a string.
- arg2 is value, which indicates the processor.
- Returns null

Example:

```
forEach(cnpdAllStatsTable,row,totalInPkts     = totalInPkts +
row.get("cnpdAllStatsInPkts"));
```

# FORMATTIME

Syntax

**FORMATTIME** (object)

Macro Description

Converts the object into an HH:MM:SS formatted string and returns the string. If a failure occurs, returns "00:00:00".

- object is the number of seconds.

Example:

```
UpTime = FormatTime(cpuUpTime/100);
```

# GENERICCSVPOLL

Syntax

**GENERICCSVPOLL** (arg1, arg2, arg3, arg4)

Macro Description

Used only in the PollDefinition section. Used specifically to retrieve Generic Bulk statistic counter values from the generated CSV files. Helps to retrieve counters that belong to the same type. Note that you need to poll all counters referenced in current and other reports with the same label in the same poll macro. This is to prevent re-parsing of bulk statistics file.

- arg1—Indicates the Generic CSV template filename as defined in /opt/CSCOppm-gw/etc/csvstats/system folder.
- arg2—List of non-key counters (comma delimited list of counter names in double quotes) to get from bulk statistics file. Each counter variable is prefixed with the data type of the counter variable so that Prime Performance Manager knows how to parse this variable.
- arg3—List of key counters (comma delimited list of counter names in double quotes) for that act as key for the counters in arg2. Each key variable is prefixed with the data type of the counter variable so that Prime Performance Manager knows how to parse this variable. If there are no key fields an empty string is used.
- arg4—Where condition clause that can be used to retrieve only selected rows that satisfy the condition expression defined.

Example:

```
pmgMsgsTable = GenericCsvPoll("pmg-perf",
                              "integer:avg response time ms,
                               integer:max response time ms,
                               integer:min response time ms,
                               integer:num msgs,
                               integer:num errors,
                               integer:total bytes received,
                               integer:total bytes sent,
                               integer:bytes received per second,
                               integer:bytes sent per second",
                              "string:msg name",
                              "(msg name == Register)");
```

# GET

Syntax

**GETALL** (object, arg1)

Macro Description

Returns the element specified by arg1.

- object is a table.
- arg1 is a string (that is a key).

Example:

```
row.get("cnpdAllStatsInPkts"));
```

# GETALL

Syntax

**GETALL** (object, arg1)

Macro Description

Returns all the column values of arg1. Returns Null in case of a failure.

- Object is a table
- arg1 is a string (that is a key).

Example:

```
cgnInstanceTable.getAll("serviceTypeNat")
```

# GETAPSTATSINFO

Syntax

GETAPSTATSINFO ()

Macro Description

This macro is used to poll the statistics of the APStats collector

**Example**:

```
getApStatsInfo();
```

# GETAVAILABILITYINFO

Syntax

**GETAVAILABILITYINFO**()

Macro Description

Gets availability information (i.e. whether the system is up or down) from the node (MWTMCURRTIME, sysUpTime, sysName, and availability). In the report in the Poll section, it needs an extra argument: alwaysExecute="true."

- Used only in the PollDefinition section.

Example:

```
deviceVariables = getAvailabilityInfo();
```

# GETCEPHCLUSTERNAME

**Syntax**

GETCEPHCLUSTER ()

Macro Description

Retrieves the Ceph cluster name as well as the Ceph cluster FSID (UUID). It returns a map with the ClusterName and FSID keys (or null in the case of failure).

Example:

```
ClusterNameTable = getCephClusterName();
```

# GETDOMAINBYIPADDRESS

Syntax

GETDOMAINBYIPADDRESS (Value ipaddress)

Macro Description

A macro to map an ip address to a top-level domain.

As a theoretical use case: maybe a customer wants a netflow report that includes a column showing all traffic related to google, youtube, gmail, and google maps simply as 'google'.

This can be achieved by listing the ip addresses (x.y.z.1) or ip address ranges (x.y.*.1-5) in the IPToDomainMap.properties file and associating them with 'google'.

- ipaddress is the IP address which is used by the macro to fetch the domain.

**Example**:

```
getDomainByIpAddress(10.10.10.3):
If IPToDomainMap.properties contains for an example, an entry such as 10.10.10.1-5 =
DomainZ. Then the macro will return DomainZ, and if such an entry does not exist, then it
will return 10.10.10.3.
```

# GETDSCP

Syntax

**GETDSCP** (arg1)

Macro Description

Used only in the ProcessPollResult section to return a textual representation of the Differentiated Services Code Point (DSCP) for input to the Type Of Service (ToS) field as arg1. The macro uses the higher order six bits of input for the ToS field for name lookup. The macro is generally used for NetFlow reports to display the name of the DSCP instead of numerical values. Table B-1 shows the DSCP name and decimal and ToS values.

***Table B-1 DSCP Decimal and ToS Values***

| DSCP Name | Decimal Value | ToS Value |
|---|---|---|
| AF11[1] | 10 | 40 |
| AF12 | 12 | 48 |
| AF13 | 14 | 56 |
| AF21 | 18 | 72 |
| AF22 | 20 | 80 |
| AF23 | 22 | 88 |
| AF31 | 26 | 104 |
| AF32 | 28 | 112 |
| AF33 | 30 | 120 |
| AF41 | 34 | 136 |
| AF42 | 36 | 144 |
| AF43 | 38 | 152 |

*Table B-1 DSCP Decimal and ToS Values*

| DSCP Name | Decimal Value | ToS Value |
|---|---|---|
| CS1[2] | 8 | 32 |
| CS2 | 16 | 64 |
| CS3 | 24 | 96 |
| CS4 | 32 | 128 |
| CS5 | 40 | 160 |
| CS6 | 48 | 192 |
| CS7 | 56 | 224 |
| EF[3] | 46 | 184 |
| default | 0 | 0 |

1. AF = assured forwarding
2. CS = class selector
3. EF = expedited forwarding

Example:

```
GetDSCP(96) returns "CS3"
```

# GETECN

Syntax

**GETECN** (arg1)

Macro Description

Used only in the ProcessPollResult section to return a textual representation of Error Congestion Notification (ECN) for input Type Of Service (ToS) field as arg1. The macro uses the lower order 2 bits of input ToS field for name lookup. Generally used for NetFlow reports to display the ECN name instead of numerical value in reports. Table B-2 shows the ToS values and ECN names.

*Table B-2 ToS Values and ECN Names*

| ToS Value | ECN Name | ECN Description |
|---|---|---|
| 00 | Not-ECT | Not ECN-Capable Transport |
| 01 | ECT(0) | ECN-Capable Transport |
| 10 | ECT(1) | ECN-Capable Transport |
| 11 | CE | Congestion Experienced |

Example:

```
GetECN(11) returns "CE"
```

# GETIPGROUP

**Syntax**

GETIPGROUP (Object ipaddress)

Macro Description

Returns an ip group given an ip address. How an ip group is defined via ip addresses is specified in IPGroupSchema, which is located under $ppm_root_dir/etc/IPGroupSchema by default.

- ipaddress is an IP address for which the macro returns an IP group.

Example:

```
getIpGroup(10.10.10.10):
The IPGroupSchema may contain an entry such as : groupA   =
10.10.10.10,192.168.0.3,10.10.11-13.5.
Then the macro would return string groupA and if it does not find a relevant entry in the
schema file, it returns a message to indicate that the group is unknown.
```

# GETHOSTADDRESS

Syntax

**GETHOSTADDRESS** (object)

Macro Description

Returns the host address in string form. Returns null in case of a failure.

- object is an IP address.
- Used only in ProcessPollResult section.

Example:

```
RserverIpAddress  = serverIpAddress.getHostAddress();
```

# GETHOSTNAME

Syntax

**GETHOSTNAME** (object, arg)

Macro Description

Returns the host name in string format. The hostname is resolved using the naming resolution defined in the optional arg parameter. If the arg parameter is not provided, the macro uses the naming resolution defined in RESOLVE_HOST_NAMES in the Reports.properties file. If RESOLVE_HOST_NAMES is not found in Reports.properties, the macro uses the DNS to resolve the IP to a hostname. Regardless of the naming resolution strategy, if the macro cannot resolve the IP address, it returns the IP address itself as the hostname.

- object is an IP address.
- arg is an optional parameter that is used to define the naming resolution. Strategy to use: DNS or PPM. The expected value is one of the following:
  - dns
  - ppm

- dns,ppm
- ppm,dns

The default is dns when nothing is provided, and the RESOLVE_HOST_NAMES setting is not found in the Reports.properties file.

The macro is used only in the ProcessPollResult section.

Example:

```
targetAddress1.getHostName()
targetAddress2.getHostName("ppm")
targetAddress2.getHostName("ppm, dns")
```

# GMONDPOLL

Syntax

GMONDPOLL (arg0, arg1, arg2)

Macro Description

Polls the remote gmond service through a specific socket port. (The default is 8649.)

- arg0 – metrics names that need to poll
- arg1 - cache key (optional)
- arg2 - true if just get local node metrics (optional, default is false)

**Example:**

```
gmondCPUStats = GmondPoll(
                  "cpu_user,
                   cpu_nice,
                   cpu_system,
                   cpu_idle,
                   cpu_wio,
                   cpu_steal,
                   cpu_intr,
                   cpu_sintr,
                   load_one,
                   load_five,
                   load_fifteen,
                   cpu_num,
                   cpu_speed",
                  "gmondCPUStats");
```

# GETNETFLOWSTATS

**Syntax**

GETNETFLOWSTATS ()

Macro Description

Gathers metrics about NetFlow collection process. Flow counts received by the collector, processed into the PPM database, and missed flows (tracked by UDP sequence) are reported.

Example:

```
nfStats = getNetFlowStats();
```

# GETPINGINFO

Syntax

**GETPINGINFO**()

Macro Description

This macro uses the Device context to obtain a node object. The node object returns a list(of map) of ping results.

Example:

```
deviceVariables = getPingInfo();
```

# GETPREFIX

Syntax

**GETPREFIX** (arg1, arg2)

Macro Description

Used only in the ProcessPollResult section to return the IP Address prefix using the IP address and subnet mask as inputs.

- arg1—Fully formatted IP Address.
- arg2—Subnet mask bits.

Example:

```
GetPrefix(10.1.1.10, 24) returns an InetAddress 10.1.1.0.
```

To get the value in string format you can use the GetHostAddress() macro passing the prefix.

```
prefixObj = GetPrefix(addr, mask);
prefixString = prefixObj.getHostAddress();
```

# GETSERVERBY PORT

Syntax

**GETSERVBYPORT** (arg1, arg2)

Macro Description

Used only in the ProcessPollResult section to return the service name for the given port number and protocol name. Uses the services file in the etc directory on the gateway and unit for the service name lookup.

- arg1—Port number
- arg2—Protocol name

Generally used with NetFlow reports to display the service or application name in reports.

Example:

```
GetServByPort(80, TCP) returns HTTP.
```

# GETSTRING

Syntax

GETSTRING (arg1, arg2, [arg3], [arg4], [arg5] )

Macro Description

Gets the substring from a string that matches the regular expression passed in. The macro takes 5 arguments of which 2 are mandatory and 3 are optional. The different parameter combination includes:

- (string, regex) -> in this form, startGrp# is 0, endGrp# is the number of capturing groups in this pattern, return value = matched string/null if no match found .
- (string, regex, startGrp#) -> in this form, the endGrp# is the number of capturing groups in this pattern, return value = matched string/null if no match found
- (string, regex, startGrp#, endGrp#) -> in this form, return value = matched string/null if no match found.
- (string, regex, startGrp#, endGrp#, retval) -> in this form, the return value = string/retVal.

Example:

```
var CWCconfig = {
        'username': prefs.getString('username'),
        'ccmcipaddress': prefs.getString('ccmcipaddress'),
        'tftpaddress': prefs.getString('tftpaddress'),
        'devicename': prefs.getString('devicename')
    };
```

# GETSYSTEMPROPERTY

Syntax

**GETSYSTEMPROPERTY**(arg1, arg2)

Macro Description

Uses arg1 to fetch the system property in the form of a string. If this fails, arg2 is used as a default return value.

- arg1 is the property key.
- arg2 is the default value.

Example:

```
IfNameFormat = GetSystemProperty("IFNAME_FORMAT", "both");
```

# GROUP

Syntax

*GROUP*(object, arg1, arg2)

Macro Description

This macro has groups of rows in a table. It is useful when used with the count macro. It returns the grouped data.

- object is the row which requires grouping.

- arg1 and arg2 are the column names that need to be grouped for a row.
- Used in cmts.xml report.

**Example:**

```
 cmStatusGroups = cdxCmtsCmStatusExtTable.group("cdxCmtsCmStatusValue,
docsIfCmtsCmStatusIndex");
```

# GETWEBQUERIES

Syntax

GETWEBQUERIES ( )

Macro Description

Fetches Web Query Statistics at the server level. The web query statistics provide information on the number of web queries made on a report at the device and network levels.

Example:

```
getWebQueries();
```

# HASGMONDPOLL

Syntax

HASGMONDPOLL (arg0, arg1)

Macro Description

Checks if data can be retrieved using the gmond service.

- arg0 - metrics names that need to poll (optional)
- arg1 - true if just get local node metrics (optional)

Example:

```
GMOND_CPU_LOCAL_CAP =
hasGmond("cpu_user,cpu_nice,cpu_system,cpu_idle,cpu_wio,cpu_intr,cpu_sintr",true);
```

# HASCAPABILITY

Syntax

**HASCAPABILITY**(arg1)

Macro Description

Checks if the Capability names listed as parameters are applicable for the given Node object.

- arg1 is the capability names.

Example:

```
if(hasCapability("MPLS_L3_VPN");
```

# HASINTERFACE

Syntax

**HASINTERFACE**(arg1)

Macro Description

Enables the setting of capabilities based on the existence of certain interfaces.

- arg1 isa conditional statement used to check if any of the device interfaces match the condition
- Used only in the SystemCapability.xml file.

Example:

```
hasInterface(ifDescr.startsWith("GigabitE"));
```

# HASMATCHINGENTRIES

Syntax

**HASMATCHINGENTRIES** (arg1,arg2)

Macro Description

Checks if the table has the entries by matching filter.

- arg1 is the poll result table
- arg2 is the matching filter

Example:

```
Y1731_MIB_IOS    = Not(IOS_XR) @and hasMatchingEntries(
                            poll("rttMonCtrlAdminIndex,
                                  rttMonCtrlAdminOwner,
                                  rttMonCtrlAdminTag,
                                  rttMonCtrlAdminRttType,
                                  rttMonCtrlAdminFrequency",
                                 "rttMonCtrlAdminTable"),
                            (rttMonCtrlAdminRttType @eq 23) @or (rttMonCtrlAdminRttType
@eq 24));
```

# HASSENSOR

Syntax

**HASSENSOR** (arg1,arg2)

Macro Description

Checks for the strings "transmit', 'receive', 'tx', 'rx.' Because these are common strings in the entity table and are not necessarily optical sensor specific, might also need to inspect the entityphysicalvendortype field.

- arg1 is the entity table
- arg2 is the matching filter

Example:

```
CISCO_ENTITY_SENSOR_MIB_RX = CISCO_ENTITY_SENSOR_MIB @and
```

```
                                    (Not(CPT_NGXP) @and
hasSensor(entPhysicalVendorType.contains("1.3.6.1.4.1.9.12.3.1.8.46")));

CISCO_ENTITY_SENSOR_MIB_TX = CISCO_ENTITY_SENSOR_MIB @and
                                    (Not(CPT_NGXP) @and
hasSensor(entPhysicalVendorType.contains("1.3.6.1.4.1.9.12.3.1.8.47")));

CISCO_ENTITY_SENSOR_MIB_TEMP = CISCO_ENTITY_SENSOR_MIB @and
                                    (Not(CPT_NGXP) @and
hasSensor(entPhysicalVendorType.contains("1.3.6.1.4.1.9.12.3.1.8.50")));

CISCO_ENTITY_SENSOR_MIB_CURRENT = CISCO_ENTITY_SENSOR_MIB @and
                                    (Not(CPT_NGXP) @and
hasSensor(entPhysicalVendorType.contains("1.3.6.1.4.1.9.12.3.1.8.48")));

CISCO_ENTITY_SENSOR_MIB_VOLTAGE = CISCO_ENTITY_SENSOR_MIB @and
                                    (Not(CPT_NGXP) @and
hasSensor(entPhysicalVendorType.contains("1.3.6.1.4.1.9.12.3.1.8.49")));
```

# HASVAR

Syntax

**HASVAR** (arg1)

Macro Description

Checks if the MIB variables in the parameter list exists for the given device.

- arg1 is either a table or an instance variable of MIB
- Used only in the SystemCapability.xml file.

Example:

```
TCP_MIB  = hasVar("tcpInSegs");
```

# HEX2STRING

Syntax

**HEX2STRING**(object, arg1)

Macro Description

Supports the HEX to string conversion. The delimiter is supported as input parameter to separate the HEX values.

- object holds the HEX values.
- arg1 is the delimiter.

Example:

```
fcIfWwn = fcIfWwn.hex2String(":");
```

# HOSTANDPLUGINPOLL

Syntax

HOSTANDPLUGINPOLL ()

Macro Description

Polls the host name and the plugins for collected

Example:

```
hostAndPluginPoll();
```

# HYPERVISORPOLL

Syntax

**HYPERVISORPOLL**(arg1, arg2, arg3, arg4, arg5)

Macro Description

Support the Hypervisor collector. It returns the results from the Hypervisor Poll.

- arg0 is the function name
- arg1 are the input parameters
- arg2 are the keys for values
- arg3 indicates static or dynamic
- arg4 is the cache key

Example:

```
hypervisorPoll("ListVMAvailability", "",  "", false);
```

# HYPERVISORPOLLPERSIST

Syntax

**HYPERVISORPOLLPERSIST**()

Macro Description

Fetches the hypervisor capability and persists it.

Example:

```
hypervisorPollPersist().
```

# IF

Syntax

**IF** (object, arg1, [arg2])

Macro Description

If two arguments are used, returns/executes arg1 if object is True or returns Null if object is False. If three arguments are used, returns/executes arg1 if object is True or returns/executes, arg2 if object is False.

Object is a Boolean type

Example:

```
 if(Linux,LinuxDevices());
```

# IFDESCR

Syntax

**IFDESCR** ([object])

Macro Description

Returns a description of the interface by looking it up first, to prevent re-polling. It returns the optional argument is the index key for the operation (default is ifIndex).

Used only in the ProcessPollResult section.

Example:

```
interfaceDescr   = IfDescr(interfaceIndex);
```

# IFINFO

Syntax

**IFINFO** ()

Macro Description

Returns a list of rows that contain Iftable and ifXtable SNMP data for the current device.

Example:

```
interfaceTable = IfInfo();
```

Maximum interface bandwidth is variable. For example, a user can configure QoS on a gigabit interface, so ifSpeed 1000 MBPS). SNMP polling might determine the interface bandwidth is 300 MBPS, so 300 MBPS is the maximum bandwidth instead of 1000 MBPS.

You can use the devicePolicies.xml file, located in /opt/CSCOppm-gw/etc/policies/users to configure specific maximum bandwidths for interfaces instead of relying on SNMP polling. Each device policy is defined in the following format:

```
    <Policy name="devicePolicies">
        <Processing>
            deviceSettings = ProcessorMap(
            {
                "Node=<FQDN>" =
                {
                    "Interfaces" =
                    {
                        "<ifDescr>" = { "ifSpeed" = <actual ifSpeed> },
                        "<ifDescr>" = { "ifSpeed" = <actual ifSpeed> },
                        ...
                    }
                },
                ..
            });
        </Processing>
    </Policy>
```

For example:

```
<ns:PolicyList xmlns:ns="http://cisco.com/ppm/poller">
    <Policy name="devicePolicies">
        <Processing>
            deviceSettings = ProcessorMap(
            {
```

```
                "Node=192.168.0.1" =
                {
                    "Interfaces" =
                    {
                        "GigabitEthernet0/0" = { "ifSpeed" = 300000000 },
                        "GigabitEthernet0/1" = { "ifSpeed" = 600000000 }

                    }
                },
    "Node=2011:0:0:162::130" =
    {
        "Interfaces" =
        {
            "GigabitEthernet0/0" = { "ifSpeed" = 1000000 }
        }
    }
        });
        </Processing>
    </Policy>
</ns:PolicyList>
```

# IFSPEED

Syntax

**IFSPEED** ([object])

Macro Description

Returns the configured interface speed.

Used only in the ProcessPollResult section.

The optional argument is the index key for the operation (default is ifIndex).

Example:

```
txSpeed = IfSpeed();
```

# IFSPEEDRECEIVE

Syntax

**IFSPEEDRECEIVE** ([object])

Macro Description

Returns the receive interface speed.

- The optional argument is the index key for the operation (default is ifIndex).

Example:

```
ifSpeedReceive();
```

# IFTABLE

Syntax

**IFTABLE** ( )

Macro Description

Returns a list of rows that contain the Iftable SNMP data for the current device.

Example:

```
interfaceTable = IfTable();
```

# INDEXOF

**Syntax**

INDEXOF (object, arg1)

Macro Description

Provides the index of a character or substring in a string.

- Object - is the string
- arg1 – is the character or substring whose index in Object needs to be obtained.

Example:

```
win = ( na.indexOf( 'Win' ) != -1 );
```

# INRANGE

Syntax

**INRANGE**(arg1, arg2)

Macro Description

Supports the range matching for MPLS OAM reports, in which it needs to check if the IPSLA index is included in the probe list rttMplsVpnMonCtrlProbeList. The following cases are supported by this macro:

(a) Individual ID's with comma separated as 1,5,3.

(b) Range form including hyphens with multiple ranges being separated by comma as 1-10,12-34.

(c) Mix of the above two forms as 1,2,4-10,12,15,19-25.

Returns true if successful and false otherwise.

- arg1 is string which presents the data range
- arg2 is a number with string format

Example:

```
InRange(rttMplsVpnMonCtrlTable.rttMplsVpnMonCtrlProbeList,
rttMonStatsCaptureTable.rttMonCtrlAdminIndex))
```

# INTERVALDURATION

Syntax

**INTERVALDURATION** ()

Macro Description

Returns the time interval for the given report (in seconds). For reports, this will typically only be 15 min, 1 hour, 1 day, and so on.

- Used only in the WebReport or CSV section.

Example:

```
IntervalDuration();
```

# INTVALUE

Syntax

**INTVALUE** (object)

Macro Description

Converts the object into an integer. Returns Null, if there is an error.

Example

```
cpwVcMplsLocalLdpID.intValue(4,2);
```

# INVENTORYPERSIST

Syntax

**INVENTORYPERSIST** ()

Macro Description

Persists the value of last change time on running config and entity for other macro usage.

Example:

```
InventoryPersist();
```

# IOSVERSION

Syntax

**IOSVERSION** (object)

Macro Description

If object is "sysDescr" (which is typical), parses the IOS version from the given string.

- Used only in the SystemCapability.xml file (only needs to be used once).

Example:

```
iosVer = iosVersion(deviceVersion);
```

# IPADDRESS

Syntax

**IPADDRESS** (object, [arg1, arg2])

Macro Description

If only one argument is used, returns the object converted into an IP address. If only two arguments are used, returns object (starting at byte "arg1") converted into an IP address. If three arguments are used, returns object (starting at byte "arg1" and ending at byte "arg2") converted into an IP address. Offsets are similar to Java's substring function.

- Object is an octet string address and arg1/arg2 are offsets.
- Used only in the ProcessPollResult section.

Example:

```
cmStatusIpAddress     = docsIfCmtsCmStatusInetAddress.ipAddress();
```

# IPADDRTOLONG

Syntax

IPADDRTOLONG ()

Macro Description

Transforms the OSPF area ID value with an IP-address-format to a long-value-format. For example, the IP address, 0.0.1.44, corresponds to 300, and 255.255.255.255 corresponds to 4294967295.

**Note** When used in the WebReport section of xml files, transform the long value to String type by calling ToString() macro, so it can be displayed correctly in the GUI. Otherwise, it might display '4.29G' for '4294967295'.

**Example:**

```
AreaIdLong   =  ospfAreaId.IPAddrToLong();
```

# ISCOLLECTD

Syntax

ISCOLLECTD ()

Macro Description

Determines whether this is a collectd.

Example:

```
isCollectd();
```

# ISINGROUP

Syntax

ISINGROUP ()

Macro Description

Checks whether a certain IP address belongs to a user specified IP group. It is mainly used to filter records and in turn only dump the filtered records into the DB to save space.

Example:

```
isInGroup();
```

# ISHYPERVISOR

Syntax

**ISHYPERVISOR** ( )

Macro Description

Determines whether the device is a hypervisor.

Example:

```
if( isHypervisor(), AllHypervisors(), AllDevices());
```

# ISNULL

Syntax

**ISNULL** (object, [arg1])

Macro Description

If one argument is used, returns True if the object is Null and False otherwise. If two arguments are used, returns the object if it is not Null or returns arg1 if it is Null.

Example:

```
txOctets = ifHCOutOctets.isNull(ifOutOctets);
```

# ISTABLEEMPTY

Syntax

**ISTABLEEMPTY** (object)

Macro Description

Checks if each MIB table in the parameter list is empty or does not exist for the node. Returns True if the object is empty.

- Object is MIB table name.
- Used only in the SystemCapability.xml file.

Example:

```
isTableEmpty("dot3HCStatsTable").
```

# ISTABLENOTEMPTY

Syntax

**ISTABLENOTEMPTY** (object)

Macro Description

Checks if each MIB table in the parameter list is not empty. Returns True if object is not empty.

- Object is a table.

Example:

```
isTableNotEmpty("mplsInSegmentTable")
```

# JOIN

Syntax

**JOIN** (object, arg1, arg2)

Macro Description

Returns the resulting joined tables of *object* and arg1. A row from *object* and a row from arg1 are joined together if the condition (arg2) is true.

- object and arg1 are tables and arg2 specifies a match condition.
- Used only in the PollDefinition section.

Example:

```
stats = stats.join(interfaceTable, (stats.ifIndex == interfaceTable.ifIndex));
```

# JMXCREDSEXIST

Syntax

JMXCREDSEXIST ()

Macro Description

Checks if the JMX Credentials exist for a particular device.

Example:

```
JMX = JMXCredsExist();
```

# JMXPOLL

Syntax

**JMXPOLL**(BeanName [Path [**s** | **c** | **d**]

Macro Description

Retrieves Java Management Extension (JMX) attributes from a server. The macro can be invoked in one of two ways:

- JMXPoll(BeanName)—Polls only the bean name, without specifying an attribute.
- JMXPoll(BeanName)—Polls only the bean name, specifyies an attribute and the depth of the attribute to poll. Options:
  - BeanName—The full name of the bean.
  - Path—The name of an attribute and its subattributes, if any. The # delimiter used between levels of an attribute, that is, between an attribute and its subattributes.
  - s—Retrieves the simple attributes at the path. Example: String, long, string array, long array.
  - c—Retrieves all the children attributes from the path. Example: composite data, tabular data.

  - c—Retrieves all the descendants, up to the leaf, from the path. Example: composite data, tabular data. Examples: composite data array ,combination of composite and tabular data.

Example:

```
JMXPoll("java.lang:type=Runtime","SystemProperties#java.ext.dirs ","s");
where 'java.lang:type=Runtime' is the beanName.
         'SystemProperties' is the attribute.
         'java.ext.dirs' is the sub-attribute.
```

The JVM bean has many data types. Here is a summary with an example of their JMXPoll invocation.

- Simple data Type

  ```
  JMXPoll("java.lang:type=Compilation","Name","s");
  ```

- String Array

  ```
  JMXPoll("java.util.logging:type=Logging","LoggerNames","s");
  ```

- Composite data

  ```
  JMXPoll("java.lang:type=MemoryPool","CollectionUsage","c");
  ```

- Tabular Data

  ```
  JMXPoll("java.lang:type=Runtime","SystemProperties","d");
  ```

- Integer Array

  ```
  JMXPoll("java.lang:type=Threading","AllThreadIds","s");
  ```

- Composite Data Array

  ```
  JMXPoll("com.sun.management:type=HotSpotDiagnostic","DiagnosticOptions","d");
  ```

- Tabular + Composite Data Array

  ```
  JMXPoll("java.lang:type=GarbageCollector","LastGcInfo","d");
  ```

# LEFTJOIN

Syntax

**LEFTJOIN** (object, arg1, arg2)

Macro Description

Returns the resulting joined tables of object and arg1. A row from object and a row from arg1 are joined together if the condition (arg2) is True. However, each row in the object continues to be retained in the resulting table even if it does not match any row from the object specified in arg1.

- Object and arg1 are tables and arg2 is a match condition.

Example:

```
casStats.leftJoin(casConf, ((casConf.casIndex == casStats.casIndex    @and
(casConf.casProtocol == casStats.casProtocol))));
```

# LEFTJOINMANY

Syntax

LEFTJOINMANY (object, arg1, arg2)

Macro Description

Returns the resulting joined tables of object and arg1. A row from object and a row from arg1 are joined together if the condition (arg2) is True. Each row from object can match with multiple rows in arg1. However, each row in the object continues to be retained in the resulting table even if it does not match any row from the object specified in arg1.

- Object and arg1 are tables and arg2 is a match condition

Example:

```
nhrpCacheTable = nhrpCacheTable.leftJoinMany(ipCidrRouteTable,
((ipCidrRouteTable.ipCidrRouteNextHop ==
IpAddress(nhrpCacheTable.nhrpCacheInternetworkAddr))));
```

# LENGTH

Syntax

**LENGTH** (object)

Macro Description

Returns the number of characters in object. It is similar to Java's length string function.

- Object is a string type.

Example:

```
if(((Length(cpwVcMplsPeerLdpID) == 6))
```

# LONGVALUE

Syntax

**LONGVALUE** (object)

Macro Description

Converts object into a long value. Returns Null, if there is an error.

Example:

```
LongValue(GetSystemProperty("IPSLA_FTP_FILE_SIZE", 1048576)), size);
```

# MAP

Syntax

MAP ()

Macro Description

Creates a Jav- like map.

Example:

```
tenantsSwiftIn = Map();
forEach(swiftInTable, row, tenantsSwiftIn.add(row.get("project_id"),
(IsNull(tenantsSwiftIn.get(row.get("project_id")), 0) + row.get("counter_volume") /
100)));
```

# MACADDRESS

Syntax

**MACADDRESS** (object)

Macro Description

Returns the specified MACAddress format defined in SNMPv2-TC.

- Object is the Byte Array format of the MACAddress.

Example:

```
flapMacAddr = ccsCmFlapMacAddr.macAddress().
```

# MATCHES

Syntax

**MATCHES** (object, arg1)

Macro Description

Returns True if the regular expression pattern arg1 is found in object.

- Object is a string and arg1 is a Java regular expression pattern.

Example:

```
ifDescr.matches("\w"));
```

# MATCHESGROUP

Syntax

MATCHESGROUP(String GroupName, String ProcessingName)

Macro Description

Tests whether the data in this context matches the given group and processing.

Example:

```
filter = "MatchesGroup(\"" + groupName + "\", \"" + processingName + "\")";
```

# NOT

Syntax

**NOT** (arg)

Macro Description

Returns the opposite of object.

- arg is a boolean type.

**Example**:

```
not( ifDescr.startsWith("unrouted vlan") );
```

# OPENSTACKPOLL

Syntax

OPENSTACKPOLL (arg0, arg1, arg2, arg3, arg4, arg5, arg6, arg7)

Macro Description

Polls meters using the OpenStack Ceilometer service.

- arg0 - package ID
- arg1 - action name
- arg2 - input parameters
- arg3 - values needs to be polled
- arg4 - keys for values
- arg5 - service type
- arg6 - static or dynamic config
- arg7 - cache key

Example:

```
vmCPUTable = OpenstackPoll("PalOS",
"PalOS.ceilometerVMCPUUtil",
"",
"octetstring:project_id,
octetstring:resource_id,
octetstring:display_name,
octetstring:name,
float:counter_volume,
octetstring:timestamp",
"resource_id",
"metering",
false,
"vmCPUTable");
```

# OSCREDSEXIST

Syntax

OSCREDSEXIST()

Macro Description

Checks whether OpenStack credentials are configured.

Example:

```
OPENSTACK_CAP = OSCredsExist();
```

# PARSESTRING

Syntax

**PARSESTRING** (arg1, arg2, arg3)

Macro Description

Parses the arg1 into tokens and returns the token identified by arg2

- arg1 is a string constant or variable.
- arg2 is a delimiter character used to separate the arg1 into tokens.
- arg3 is an integer number indicating which token to return.

Example:

```
chassisName = ParseString(adaptorDesc, "/", 2);
ParseString("sys/chassis-1/blade-2/board/memarray-1", "/", 2) returns "chassis-1".
```

# POLL

Syntax

**POLL** (arg1, arg2)

Macro Description

Polls only the group of scalar values or table variables that share the same index.

- Used only in the PollDefinition section.
- arg1 is a list of variables (comma delimited list of variables in double quotes) to poll
- arg2 is the label used to allow you to reference it in other polls to prevent re-polling..

**Note** You must poll all variables referenced in current and other reports with the same label in the same poll macro. This is to prevent missing variables. Once a poll result is associated with the label the device will not be re-polled when the poll macro is called with the same label.

Example:

```
cpmCPUTotalTable = poll("cpmCPUTotalIndex,
cpmCPUTotalPhysicalIndex,
cpmCPUTotal5minRev,
cpmCPUTotal1minRev");
```

# POLLMT

Syntax

POLLMT (String keys, String, String)

Macro Description

Polls the MediaTrace mib table.

Example:

```
cMTCommonMetricStatsTable = pollMT("cMTSessionNumber,
cMTSessionLifeNumber,
cMTHopStatsAddrType,
cMTHopStatsAddr",
"cMTCommonMetricsFlowSamplingStartTime",
"cMTCommonMetricsFlowSamplingStartTime,
cMTCommonMetricsIpPktDropped,
cMTCommonMetricsIpOctets,
cMTCommonMetricsIpPktCount,
cMTCommonMetricsIpByteRate");
```

# POLLNEXT

Syntax

**POLLNEXT** (arg1, arg2, arg3, arg4)

Macro Description

Retrieves the data that you have not already retrieved (retrieves the next available data).

Used only in the PollDefinition section.

- arg1 is the index that uniquely identifies each row (comma delimited list of variables in double quotes)
- arg2 is the list of variables to poll (comma delimited list of variables in double quotes)
- arg3 is list of index variables to poll
- arg4 is a Boolean value that is true if you want to return the last row even if it is the same as the last row of the last poll.

**Example**:

```
rttMonStatsCaptureTable =
pollNext("rttMonCtrlAdminIndex,
rttMonStatsCapturePathIndex,
rttMonStatsCaptureHopIndex,
rttMonStatsCaptureDistIndex",
"rttMonStatsCaptureStartTimeIndex",
"rttMonCtrlAdminIndex,
rttMonStatsCaptureStartTimeIndex,
rttMonStatsCapturePathIndex,
rttMonStatsCaptureHopIndex,
rttMonStatsCaptureDistIndex,
rttMonStatsCaptureCompletions,
rttMonStatsCaptureSumCompletionTime",
true);
```

# POLLPERSIST

Syntax

**POLLPERSIST** (arg1)

Macro Description

Polls the variables in the list and persists the values (in row map) to make the values available in other algorithms.

Polls the values and puts it into the current context (available for immediate use in other capability checks). Used only in the SystemCapability.xml file

- arg1 is a list of variables to poll.

Example:

```
VARS = pollPersist("sysDescr,sysObjectID");
```

# PRINT

Syntax

**PRINT** ([object])

Macro Description

Prints the string version of the object in the console log.

- If no arguments are given, prints a blank line in the console log (for debugging).
- If one argument is used, prints the object in the console log (for debugging).

**Example**:

```
print(result);
```

# PROCESSORLIST

Syntax

**PROCESSORLIST** (arg1)

Macro Description

Creates Lists of objects to be referenced by internal processing code.

- arg1 is  a list of objects specified in Java Array syntax that is [ "object1", "object2", "object3" ]
- Typically used in persisted group definition files for the Usage and Objects directives.

Example:

```
Usage = ProcessorList( [ "AGG_GGSN_APN_INS_MIS", "AGG_GGSN_APN_DHCP",
"AGG_GGSN_APN_INS_PDP"] );
```

# PROTOCOLNAME

Syntax

PROTOCOLNAME (arg1)

Macro Description

Used only in the ProcessPollResult section to return the name of protocol provided in the protocol number as arg1. For example, returns UDP for 17. The macro is used in the netflow-config.xml file in the etc directory on gateways and units for protocol name lookup. The macro is generally used for NetFlow reports to display the name of protocol in reports instead of numbers.

Example:

```
protocol = ProtocolName(17);
```

# RATE

Syntax

**RATE** (object, [arg1, arg2])

Macro Description

If one argument is used, returns the rate of change between the previous and the current polling for the object (uses sysUpTime as time delay to perform calculation).

If three arguments are used (there is no two argument option), returns the rate of change between the previous and the current polling for the object. In this case, arg1 is the value used for time and arg2 is a multiplier/conversion factor in order to get the correct metric. that is, seconds or milliseconds.

For **example**, the variable sysUpTime needs a multiplier of 100 to get into seconds since it is recorded in 1/100 seconds.

Example:

```
txBitRate = txOctets.rate() * 8;
```

# REMOVE

Syntax

REMOVE(Object, arg)

Macro Description

Removes an element from a map or collection.

- Object – is the collection from which an element needs to be removed.
- arg – is the element which needs to be removed from the collection.

**Example**:

```
forEach(entPhysicalTable,entphys,entphys.remove("sysUpTime"));
```

# RUNCMD

Syntax

RUNCMD (arg0, arg1, arg2, arg3, arg4, arg5, arg6, arg7)

Macro Description

Supports script collector. Output format should be csv for now with self-defined delimiter.

- arg0 - script absolute path
- arg1 - cache key
- arg2 - delimeter for csv output
- arg3 - timeout of script processing
- arg4 - true if script will provide current timestamp
- arg5 - metrics name for csv columns
- col name like sysUpTime,MWTMCURTIME is for time setting
- arg6~N - parameters for script(optional)

### Examples

An example of the RunCmd macro used to write reports is shown below. An addition sample is provided in runCmdSampleScript located in at /opt/CSCOppm-gw/samples/runcmd/.

```
Results = RunCmd("/path/to/the/script.sh",
```

```
"thisCache",
'|',
5,
false,
"cpu_slot,
cpu_idle,
cpu_user,
cpu_nice,
cpu_system",
DeviceName(),
            "secondParm",
            "NthParm");
```

Where:

- "/path/to/the/script.sh" is the path to the script to run
- "thisCache" is the name to casche the results under in case another report wants the same data
- '|' is the delimiter for the CSV data returned by the script
- 5 is the timeout value or length of time to wait for the script to complete after which we will kill the process and continue with null results
- False indicates true or false whether PPM provides time stamps for the data or the script includes it in the csv output
- "cpu_idle, cpu_user, cpu_nice, cpu_system" is the name of the variables returned in the script csv file
- DeviceName() is the first parameter to be passed to the script
- "secondParm" is the second parameter to be passed to the script
- "NthParm" is the Nth parameter to be passed to the script

The script returns data in the following format:

```
0|1|1|1|1
1|1|2|1|1
2|1|1|3|1
3|1|1|1|4
```

This indicates four data rows are to be parsed and processed by the rest of the Prime Performance Manager framework. The macro would be written as:

```
RunCmd("/path/to/the/script.sh",
"thisCache",
'|',
5,
false,
"cpu_slot,
cpu_idle,
cpu_user,
cpu_nice,
cpu_system",
DeviceName(),
            "secondParm",
            "NthParm");
```

If the data includes the current timestamp, it would appear as:

```
0|1|1|1|1|1386142066
1|1|2|1|1|1386142066
2|1|1|3|1|1386142066
3|1|1|1|4|1386142066
```

The macro would be written as:

```
RunCmd("/path/to/the/script.sh",
"thisCache",
'|',
5,
true,
"cpu_slot,
cpu_idle,
cpu_user,
cpu_nice,
cpu_system,
MWTMCURRTIME",
DeviceName(),
            "secondParm",
            "NthParm");
```

The fourth macro parameter indicates whether the script output should provide timestamp. If set to true, the following rules apply:

- CSV timestamps should be measured in milliseconds between the current time and midnight, January 1, 1970 UTC.
- If a customer wants to define the timestamp, the CSV output should provide the current timestamp. The macro should be labeled as "MWTMCURRTIME" in the sixth parameter, which is reserved name for the current timestamp.
- If CSV output provides the system up time, you can label it as "sysUpTime" in the sixth parameter. This is an optional label. It is not necessary if the output only provides the current timestamp. The sysUpTime unit should be hundredths of a second.

# SETALGORITHMS

Syntax

**SETALGORITHMS** (arg1, arg2)

Macro Description

Defines an algorithm to be run.

- arg1 is the algorithm name.
- arg2 is either a list containing macro statements or a map of key=value pairs, where the key is the name of an object assigned the statement defined by value.

**Example**:

```
setAlgorithms("AllHypervisors", {HYPERVISOR_CAPABILITY = hypervisorPollPersist();});
```

# SETCPUINFO

Syntax

**SETCPUINFO** ([object])

Macro Description

Sets the CPU information (cpuDescr, cpuNum, cpuSlot). To override the CPU index, set object to that value.

**Example**:

```
setCpuInfo();
```

# SETMEMORYPOOLINFO

Syntax

**SETMEMORYPOOLINFO** ()

Macro Description

Creates a combined memory pool name after poll the CISCO-ENHANCED-MEMORYPOOL-MIB and ENTITY-MIB. The new memory pool name will be "slotNumber-slotName-memoryPoolName". After the data is polled, the data will be persisted for re-use if entity isn't changed and will also be put into cache for other macro use.

**Example**:

```
memoryPoolInfo = SetMemoryPoolInfo();
```

# SETTIMEVARINFO

**Syntax**

**SETTIMEVARINFO** (object, arg1, arg2)

**Macro Description**

Sets the time variable information with the given arguments in the context.

- object is the time variable to use
- arg1 is a boolean whether to use the variable next time
- arg2 is the index in which you use until it changes value.

**Example**:

```
setTimeVarInfo("sysUpTime", true);
```

# SMIEXIST

Syntax

SMIExist(arg1, arg2)

Macro Description

Checks if the SMI-S namespace or classname exist for the given device.

- arg1is a SMI-S namespace
- arg2 is a SMI-S classname

Example

```
SMIExist("emc/cimnas");
```

# SMIPOLL

Syntax

SMIPoll(arg1, arg2, arg3, arg4, arg5)

Macro Description

Used only in the PollDefinition section to retrieve CLI based report data.

- arg1 is a namespace for SMI-S polling.
- arg2 is a classname for SMI-S polling.
- arg3 is a input parameters for SMI-S polling.
- arg4 keys for table indices values.
- arg5 cache key.

Example

```
SMIPoll("emc/cimnas",
        "CIMNAS_Volume",
        "id,
        name,
        diskType,
        storageMB,
        offsetMB,
        usedMB,
        type",
        "id");
```

# STARTSWITH

Syntax

**STARTSWITH** (object, arg1)

Macro Description

Returns true if the string object starts with the string arg1

- object is a string type and arg1 is a string..
- Similar to Java's startsWith string function.

Example

```
status = ifDescr.startsWith("unrouted vlan");
```

# SYSTIME

Syntax

**SYSTIME**()

Macro Description

Returns the current system time (in milliseconds).

**Example**:

```
now = systime();
```

# TABLEINDICES

Syntax

**TABLEINDICES** (object)

Macro Description

Sets object as the value to use to identify a row in a table (comma delimited list of variables in double quotes).

**Example**:

```
tableIndices("serviceTypeNat");
```

# TOCIDSHEALTHSECMONMISSEDPKT

Syntax

TOCIDSHEALTHSECMONMISSEDPKT ()

Macro Description

This class converts the display string to the measures

Example:

```
CiscoCIDSTable = CiscoCIDSTable.ToCIDSHealthSecMonMissedPkt();
```

# TOLOWERSTRING

Syntax

**TOLOWERSTRING** (object)

Macro Description

Returns object in string form all in lower case

**Example**:

```
ifDescr6 = ifDescr6.ToLowerString(ifDescr6);
```

# TOPN

Syntax

**TOPN** (object, arg1, arg2)

Macro Description

Returns the top n (arg2) rows from object sorted by a sort key descending

- Used in the PollDefinition or ProcessDBSummary section.
- object is a table
- arg1 is what column to sort by
- arg2 is the number of records to get (n)..
- When used in the Filter section in a ProcessDBSummary section, rows is an implicit variable that typically can be used for an object that is set when ProcessDBSummary execution is complete.

**Example**:

```
rows = rows.topN("ProcessCPUUtil5mAvg", 5);
```

# TOSTRING

Syntax

**TOSTRING** (object)

Macro Description

Returns object in string form.

**Example**:

```
probeIndex = "IPSLA " + ToString(rttMonCtrlAdminIndex);
```

# TOUPPERSTRING

Syntax

**TOPUPPERSTRING** (object)

Macro Description

Returns object in string form all in upper case.

**Example**:

```
ifDescr6 = ifDescr6.ToUpperString(ifDescr6);
```

# VIEWDESCENDANT

Syntax

**VIEWDESCENDANT** (arg1)

Macro Description

Tests if a node is a descendant of the given view name. It returns true if it is a descendant; otherwise, returns false

- arg1 is the view name in the form of: persistedViewlFileName. persistedViewlFileName.viewname.subviewname
- The persistedViewFileName will vary based on whether user access is enabled or not. When not enabled the first 2 level of the view name are the device where the view was originally created. When enabled it is the id of the user.
- Typically used in group and threshold definitions to limit the scope of devices .

**Example**:

```
Algorithm =
If(ViewDescendant("dhcp-64-102-86-126-cisco-com_._dhcp-64-102-86-126-cisco-com_._andy"),
true, false)
```

# XMLPOLL

Syntax

**XMLPOLL** (arg1, arg2, arg3, arg4, arg5, arg6)

Macro Description

Used only in the PollDefinition section to retrieve CLI based report data.

- arg1 package ID for PAL call.
- arg2 action name for PAL call.
- arg3 input parameters for PAL call.
- arg4 values to be polled for PAL call.
- arg5 keys for table indices values.
- arg6 a static or dynamic config; indicates whether results should be cached.
- arg7 cache key

**Example**:

```
XmlPoll("y1731Stats",
"y1731Stats.y1731Config",
"1001",
"integer:rttMonCtrlAdminIndex,
octetstring:ipslaType,
octetstring:operationType,
octetstring:ipslaFrameType,
octetstring:cfmDomain,
integer:evc,
integer:cfmTargetMpid,
integer:cfmSourceMpid,
octetstring:targetMACaddress,
octetstring:sourceMACaddress,
integer:cos,
octetstring:clock",
"rttMonCtrlAdminIndex",
true,"xmlY1731ConfigTable");
```

# XMLPOLLNEXT

Syntax

**XMLPOLLNEXT** (arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8)

Macro Description

Used only in the PollDefinition section to retrieve CLI report data based on sub key.

- arg1 package ID for PAL call.
- arg2 action name for PAL call.
- arg3 input parameters for PAL call.
- arg4 values needs to be polled for PAL call.
- arg5 main keys for table indices values.
- arg6 subkeys for values for table indices.
- arg7 key for start time adjustment.

- arg8 cache key.

**Example**:

```
XmlPollNext("y1731Stats",
"y1731Stats.y1731Hist",
"1001",
"integer:rttMonCtrlAdminIndex,
datetime:startTime,
datetime:endTime,
gauge:operationInitiatedNum,
gauge:operationCompletedNum,
gauge:delayTwoWayNum,
gauge:delayTwoWayMin,
gauge:delayTwoWayAvg,
gauge:delayTwoWayMax,
gauge:delayVarianceTwoWayPosNum,
gauge:delayVarianceTwoWayMinPos,
gauge:delayVarianceTwoWayAvgPos,
gauge:delayVarianceTwoWayMaxPos,
gauge:delayVarianceTwoWayNegNum,
gauge:delayVarianceTwoWayMinNeg,
gauge:delayVarianceTwoWayAvgNeg,
gauge:delayVarianceTwoWayMaxNeg,
gauge:delayForwardNum,
gauge:delayForwardMin,
gauge:delayForwardAvg,
gauge:delayForwardMax,
gauge:delayVarianceForwardPosNum,
gauge:delayVarianceForwardMinPos,
gauge:delayVarianceForwardAvgPos,
gauge:delayVarianceForwardMaxPos,
gauge:delayVarianceForwardNegNum,
gauge:delayVarianceForwardMinNeg,
gauge:delayVarianceForwardAvgNeg,
gauge:delayVarianceForwardMaxNeg,
gauge:delayBackwardNum,
gauge:delayBackwardMin,
gauge:delayBackwardAvg,
gauge:delayBackwardMax,
gauge:delayVarianceBackwardPosNum,
gauge:delayVarianceBackwardMinPos,
gauge:delayVarianceBackwardAvgPos,
gauge:delayVarianceBackwardMaxPos,
gauge:delayVarianceBackwardNegNum,
gauge:delayVarianceBackwardMinNeg,
gauge:delayVarianceBackwardAvgNeg,
gauge:delayVarianceBackwardMaxNeg,
gauge:lossForwardNum,
gauge:lossForwardAvailableNum,
gauge:lossForwardUnavailableNum,
gauge:lossForwardTxFrms,
gauge:lossForwardRxFrms,
percent:lossForwardMinFLR,
float:lossForwardAvgFLR,
percent:lossForwardMaxFLR,
float:lossForwardCumFLR,
gauge:lossBackwardNum,
gauge:lossBackwardAvailableNum,
gauge:lossBackwardUnavailableNum,
gauge:lossBackwardTxFrms,
gauge:lossBackwardRxFrms,
percent:lossBackwardMinFLR,
float:lossBackwardAvgFLR,
```

```
percent:lossBackwardMaxFLR,
float:lossBackwardCumFLR",
"rttMonCtrlAdminIndex",
"endTime",
"startTime",
"xmlY1731StatsTable");
```

# XMLPOLLPERSIST

Syntax

**XMLPOLLPERSIST** (arg1)

Macro Description

Designed for system capability detection by the CLI poll. It is used only in SystemCapability.xml to detect the CLI poll capability(remove).

- arg1 the capability name defined in etc/palRuntime/conf/DeviceCapability.xml(add).

**Example**:

```
BGP4_SUMMARY = xmlPollPersist("bgpIpv4Summary");
```

# XMLPOLLTKPERSIST

Syntax

XMLPOLLTKPERSIST (arg1)

Macro Description

This macro is used for checking the xml capability. It extends XmlPollPersist and adds logic to get the token before polling the device.

- arg1 – is the capability name defined in etc/palRuntime/conf/DeviceCapability.xml(add).

Example:

```
UCS_C_SYS_CAP = xmlPollTKPersist("ucsCapability");
```

# XMLPOLLWITHTOKEN

Syntax

XMLPOLLWITHTOKEN (arg1,arg2, arg3, arg4, arg5, arg6, arg7)

Macro Description

Supports the XML collector and is an extension of XmlPoll. It provides a token check before polling device.

- arg1 package ID for PAL call.
- arg2 action name for PAL call.
- arg3 input parameters for PAL call.
- arg4 values to be polled for PAL call.
- arg5 keys for table indices values.
- arg6 a static or dynamic config; indicates whether results should be cached.

- arg7 cache key

Example:

```
memoryUnitEnvStatsTable =  XmlPollWithToken("UCSCIMC",
"UCSCIMC.memoryUnitEnvStats",
"",
"octetstring:dn,
octetstring:description,
octetstring:temperature",
"dn",
false,
"memoryUnitEnvStatsTable");
```

# Y1731XMLPOLLNEXT

Syntax

Y1731XMLPOLLNEXT (arg0, arg1, arg2, arg3, arg4, arg5, arg6, arg7, arg8, arg9)

Macro Description

Y1731XmlPollNext is used to poll the Y1731 data from IOX device.

- arg0 - package ID
- arg1 - action name
- arg2 - input parameters which contains profile ID and measurement name
- arg3 - values needs to be polled
- arg4 - main keys for values
- arg5 - sub keys for values
- arg6 - the name of the start time ticket
- arg7 - cache key
- arg8 - the persist flag
- arg9 - the time when run the macro

Example:

```
Y1731XmlPollNext("y1731Stats",
"y1731Stats.y1731HistIox",
row.getAll("profile,measures"),
"octetstring:profileName,
datetime:startTime,
timewithunit:endTime,
datetime:currTime,
octetstring:y1731Type,
octetstring:packetType,
octetstring:source,
octetstring:destination,
octetstring:cfmDomain,
gauge:operationInitedNum,
gauge:operationLostNum,
gauge:operationCorruptNum,
gauge:operationMisorderNum,
gauge:operationDuplicateNum,
boolean:roundTripDelay,
boolean:delayForward,
boolean:delayBackward,
boolean:delayVarianceTwoWay,
```

```
boolean:delayVarianceForward,
boolean:delayVarianceBackward,
boolean:slmForward,
boolean:slmBackward,
floattimewithunit:operationMax,
floattimewithunit:operationMin,
floattimewithunit:operationAvg",
"source,destination,cfmDomain,profileName,y1731Type",
"endTime",
"startTime",
"xmlY1731StatsTable",
true,
"currTime"))
```