Cisco Prime Performance Manager 1.0 Integration Developer's Guide

June 2011

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-24598-01

Cisco Prime Performance Manager 1.0 Integration Developer's Guide

CONTENTS

INDEX

Preface

This preface describes the objectives, audience, organization, and conventions of the *Cisco Prime Performance Manager 1.0 Integration Developer's Guide*. It refers you to related publications and describes online sources of technical information.

Cisco Prime Performance Manager is a web-based software application that enables a wide variety of service reports in your managed network, including device availability reports, SLA reports, protocol performance reports, and so on. The Prime Performance Manager gateway provides a selection of report XML files that you can use as models for creating your own reports.

This guide describes how to use the sample reports provided with Cisco Prime Performance Manager to develop your own XML reports.

This preface includes:

Objectives

This guide describes how to use the sample reports provided with Cisco Prime Performance Manager to develop your own XML reports.

Audience

This guide is for software developers who use the Cisco Prime Performance Manager and XML reports to develop customized reports. Developers should have:

- Basic network management skills
- Basic multicast knowledge
- Knowledge of Java, SNMP concepts, and XML

Organization

This guide contains the following chapters:

- Chapter 1, "Getting Started with Cisco Performance Management Reports" provides an overview of the Prime Performance Manager reports.
- (Chapter 2, "Writing a Report," provides a tutorial on report writing, based on one of the reports provided with Cisco Prime Performance Manager.
- Chapter 3, "Testing and Debugging Your Report," explains how to debug your report and lists error messages that might appear.
- Chapter 4, "XML Reference" provides reference information on the XML elements provided.
- Chapter 5, "Reports Macro Reference," provides reference information on the reports macros used with Prime Performance Manager reports.

Conventions

This guide uses basic conventions to represent text and table information.

Item	Convention
Commands and keywords	**boldface** font
Variables for which you supply values	*italic* font
Displayed session and system information	`screen` font
Elements that are optional	Square brackets ([ ])
Alternate but required keywords that are grouped	Braces ({ }) and separated by a vertical bar (|)
Information you enter	**`boldface screen`** `font`
Variables you enter	*`italic screen`* font
Menu items and button names	**boldface** font
Selecting a menu item in paragraphs	**Option > Network Preferences**
Selecting a menu item in tables	Option > Network Preferences

Examples use the following conventions:

- Terminal sessions and information that the system displays are printed in `screen` font.
- Information that you enter is in **`boldface screen`** font. Variables for which you enter actual data are printed in *italic screen* font.
- Nonprinting characters, such as passwords, are shown in angle brackets (< >).
- Information that the system displays is in `screen` font, with default responses in square brackets ([ ]).

This publication also uses the following conventions:

- Menu items and button names are in **boldface** font.
- If items such as buttons or menu options are dimmed on the application window, it means that the items are not available either because you do not have the correct permissions or because the item is not applicable at this time.

Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in the manual.

Tip Means *the following are useful tips*.

Related Documentation

Additional information can be found in the following publications of the Cisco Prime Performance Manager documentation set:

- *Cisco Prime Performance Manager 1.0 User Guide*
- *Cisco Prime Performance Manager 1.0 Quick Start Guide*
- *Cisco Prime Performance Manager 1.0 Release Notes*
- *Cisco Prime Performance Manager 1.0 Documentation List*

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.

CHAPTER 1

Getting Started with Cisco Performance Management Reports

Cisco Prime Performance Manager is a highly scalable and easy to use performance management system. It allows Service Providers to proactively manage their next generation networks, including service assurance and capacity planning.

This guide describes how to extend the built-in reports that come “out of the box” with Cisco Prime Performance Manager by developing your own customized service reports.

This chapter contains:

XML-Based Service Reports

Cisco Prime Performance Manager reports are coded in XML. Out-of-the-box, Cisco Prime Performance Manager includes more than 200 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
- Availability of device interfaces, MPLS tunnels, pseudowires, and SNMP devices
- IP Quality of Service (QoS)
- IP Protocol performance
- IP Service-Level Agreement (SLA) statistics
- Resource Utilization, such as CPU and memory utilization
- Transport Statistics, such as ATM statistics, Ethernet Virtual Circuit (EVC) statistics, and so on

For details, see the *Cisco Prime Performance Manager 1.0 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 140 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 Cisco 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 30 XML report files that you can use as a model for your own reports.
- **Properties Files**—Each XML report file has as 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 (Cisco ANA) 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
MIBs	/opt/CSCOppm-gw/etc/mibs
Capability Files	/opt/CSCOppm-gw/etc/ SystemCapability.xml /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

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

Files	Location
BQL Files	/opt/CSCOppm-gw/etc/bql/xl
Report Macros	Precompiled on the Cisco Prime Performance Manager gateway. See Chapter 5, "Reports Macro Reference" for reference information on the macros.
Event poller schema	The main XML schema for event polling is EventPoller.xsd.It is located in the/opt/CSCOppm-gw/etc/poller directory. You may view this file for reference, but do not edit it.

How It Works

The Cisco Prime Performance Manager processes reports as follows:

1. Cisco Prime Performance 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 Cisco 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, Cisco 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, Cisco 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 Cisco 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 Cisco 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

Cisco 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, Cisco 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 on 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 Frame Source Menu For Color Coded View)

Custom Help
================================================

MIB Used: CISCO-PROCESS-MIB.my
   CISCO-PROCESS-MIB.my

MIB Used: ENTITY-MIB.my
   ENTITY-MIB.my

MIB Variables Polled:


      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));
```

```
================================================================================
CSV File Format
Report ID: CPU                                  MIB Used: CISCO_PROCESS_MIB
================================================================================

CSV Filename Prefix: CPU
--------------------------------------------------------------------------------
 1 TimeStamp                                TimeStamp
 2 Node                                     fqdnid
 3 Slot                                     CPUSlot
 4 Number                                   CPUNum
 5 Description                              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

================================================================================
Web Reports
Report ID: CPU                                  MIB Used: CISCO_PROCESS_MIB
================================================================================

1 CPU Utilization
--------------------------------------------------------------------------------
Average Utilization                     Avg(cpmCPUTotal5minRev / 100)
Peak Utilization                        Max(cpmCPUTotal5minRev / 100)
Node                                    fqdnid
Slot                                    CPUSlot
CPU                                     CPUNum
CPU Description                         CPUDescr
Avg                                     Avg(cpmCPUTotal1minRev / 100)
Peak                                    Max(cpmCPUTotal1minRev / 100)
Rising                                  cpmCPURisingThresholdValue / 100
Falling                                 cpmCPUFallingThresholdValue / 100
```

Reports Help Page

To display the online reports help, select **Home > Reports Navigation** from the main menu. The online Reports Documentation includes:

- System Reports README - MIBs, Poll Definitions, and CSV Formats.
- User Reports README - MIBs, Poll Definitions, CSV Formats
- Report XML Definitions
- 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 provide code reports to provide users with:

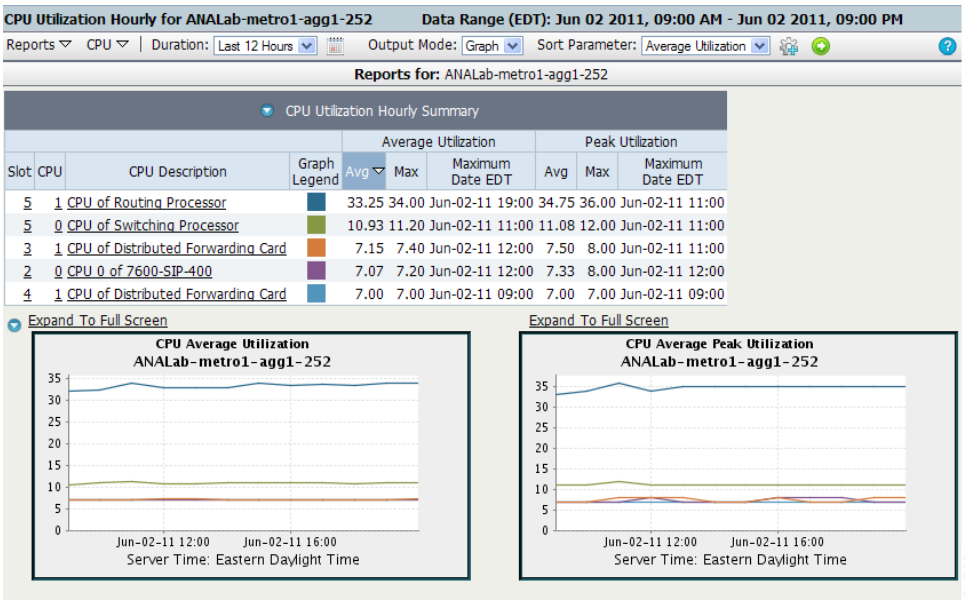
- **Graph Views**—Display a graph of performance over specified time intervals.
- **Table Views**—Display performance in a tabular view.
- **CSV File View**—Lets users save the report in a comma-separated value (CSV) file that they can view using a spreadsheet or a text editor.

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 is an example of a Graph View for CPU Utilization.

Figure 1-1 Graph View for CPU Utilization Report

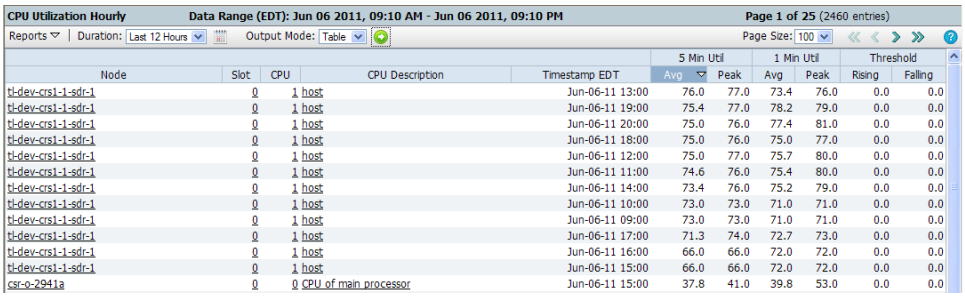


When you code your own report, you can control the time intervals that users can select.

Table View

Figure 1-2 is an example of a table view for the CPU Utilization report.

Figure 1-2 Table View



CSV File View

If users select the CSV file view, they are prompted to save the report as a CSV file. The CSV file can be viewed in a spreadsheet such as Microsoft Excel or using a text editor.

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

- **Reports Status Table**—In the Prime Performance Manager user interface, select **Reports** and click the Reports Status tab to display the Reports Status table. This allows you to enable or disable reports.
- **Reports Settings Page**—Choose **Reports** and click the Settings tab to enable various report intervals and control report aging.

For details on the user interface, see the *Cisco Prime Performance Manager 1.0 User Guide*.

CHAPTER 2

Writing a Report

Writing a report for Cisco Prime Performance Manager requires careful planning and following the report writing steps in the correct order. In this chapter, we walk you though 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

You should:

- Do initial development of a new report XML file offline from any server until the basic structure of the report file is ready. Then deploy it to the staging server for initial testing and debugging.
- Avoid developing reports on a live system. Use a staging system for developing new reports and then, once the report is verified to be working correctly, deploy it to your production servers.

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 140 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, go to the home page and 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.

Warning **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.

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, *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 2-3 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, and then click on the report help icon: 

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/mwtm/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/mwtm.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 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 *SystemCability.xml* file or the *UserCapability.xml* file.

  See Specify the MIB Criteria (Criteria Section), page 2-7.

- **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 2-9.

- **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 2-10.

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 select the CSV view option.

  See Specify the CSV Output File and Format (CSV Section), page 2-11.

- **WebReport**—Specifies the attributes of the graphical report that users see when they select the Graph View option. 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 2-12.
  - **GraphView**—Specifies the attributes of the graph view that users see when they select the Graph View option.

    See Coding the GraphView Section, page 2-14.
  - **TableView**—Specifies the attributes of the table view that users see when they select the Table view option.

    See Coding the TableView Section, page 2-15.

Reporting Macros

You can use the reports macros provided with the Prime Performance Manager gateway in your XML code. Some of the reports macros can be used only in specific sections of the XML. For reference information on the reports macros, see Chapter 5, "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

-->
```

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.

Specify the Poll Name and Report Name (Poll Element)

The Poll element:

- Specifies the name of the poller and the Report ID, for example:

  ```
  name="CPU"  reportId="CPU"
  ```

  Make sure to assign a unique name for the poller and the report ID.

  For example, if you copy the cpu.xml file 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)

Now we will look at a critical part of the sample report code—the PollDefinition section. In this section, we set up the actual polling that is done for the report and assign the polled data to internal database tables that Prime Performance Manager users to generate the reports.

Example 2-2 shows the PollDefinition section of our 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:

- cpmCPMTotalTable
- 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 cpmCPUThreshold class. 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, BUT each row in the object will still be retained in the resulting table even if it does not match with any row from the object specified in *arg1*.

Specify Poll Processing Results (ProcessPollResult Section)

The ProcessPollResult section sets up variables to manipulate the data and process it to obtain the information that we want to display in 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 rows in our virtual data table that put the data in the order it will be shown in our reports.

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 code used in our cpu.xml report is actually modeled on a "boilerplate" ProcessDBSummary section that is defined in the *process.xml* file. You can copy code from the *process.xml* file into other reports that display CPU process information, but do not modify the *process.xml* file.

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>
```

The polling data and the values calculated from it are retained in the virtual table in memory so as to be available for any CPU reports that users select in the Prime Performance Manager user interface.

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.

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—the CSV section and the WebReport section.

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 uses select the CSV report option.

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 three values:

- The name of the report.
- The location of the report, in this case the Prime Performance manager gateway, and with the listen variable, which poller is being used.

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 (WebReport 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 2-12.

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.
- **category**—Specifies where the report appears in the Prime Performance Manager report tree.

  The line `category="level1Resources,level2CPU"` specifies that the CPU report will appear under the Resources report category, and will be identified by the text "CPU."
- **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 sort order of the report within the category to which it is assigned.

  The line `sortWeight=2` specifies that the CPU report is the second report on the Resources menu.

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.

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 graph view for the report. This includes things such as:

- The level in the reports tree where the report is shown
- The report ID
- The contexts in which the report can be viewed
- The text display in the report
- The order in which the report is listed in master lists of reports

You should specify the text that is displayed using variables defined in the properties file for the report, as shown in Example 2-8.

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 GraphsSummary 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 With Prime Performance Manager 1.0.2, 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.
- 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.

Coding a LeafGraph Section

When your report set-up drills down through the available sub-reports until the lowest level is reached, and there is only one instance left of an object that you are reporting on, you can combine different data metrics for that object in one graph. You do this by coding a LeafGraph section.

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.”

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.

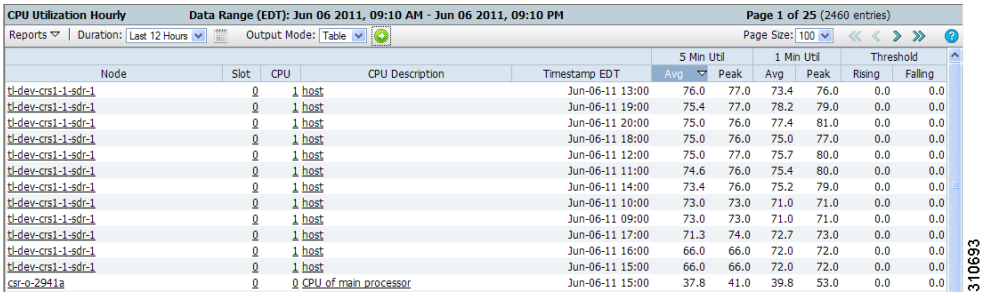
Example 2-9 TableView Section

```
<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.

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

This section discuss some best practices to follow as you develop reports.

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.

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 eliminates errors that can result from coding the same variables more than once.

Optimize Data Computations for Storage in the Database

As you develop your reports, optimize the data computations used in your reports. This saves system processing time.

Task Reference

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

Enabling 5 Minute Reports

Five minute reports are not enabled by default. Enabling 5-minute reporting is done by editing the XML report definition file that corresponds to a report. The ProcessDBSummary section and the WebReport section require the addition of interval attributes.

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

The 5-minute interval is defined as `interval="Min5Min15MinHourlyDaily"`. This attribute is not defined by default and implicitly defaults to 15 minute, hourly, and daily reports.

Note that 5-minute reporting must also be enabled globally. This is done through the web UI by an administrative user, on the Reports Settings page. The SNMP polling interval for devices from which 5-minute reports are obtained needs to be set by an administrator in the web UI on the SNMP tab (add reference to that page here).

Note Enabling 5 minute reports will significantly increase the amount of resources required for your PPM units.

Dropping Tables and Views

During report development, tables can become corrupted or have their definitions changed. Therefore it might be necessary to get rid of a database table. It might also be necessary to drop an entire SQL view.

In Cisco Prime Performance Manager, you can use SQL commands to drop tables or views.

To drop tables and views on the gateway:

Step 1 Execute the following SQL commands at: `http://<machine_name>:4440/ppm/jsp/dbquery.jsp`:

```
Drop view stats_tablename_15min
Drop view stats_tablename_hourly
Drop view stats_tablename_daily
Drop function sel_tablename
```

Step 2 On the Unit device, execute the following sql commands at:
`http://<machine_name>:5440/ppm/jsp/dbquery.jsp:`

```
Drop table stats_tablename_15min_timestamp
Drop table stats_tablename_hourly_timestamp
Drop table stats_tablename_daily_timestamp

Drop view stats_tablename_15min
Drop view stats_tablename_hourly
Drop view stats_tablename_daily
Drop function sel_tablename
Drop function bulk_ins_tablename
```

Adding a New Column to an Existing Report

To add new columns to an existing report:

Step 1 Perform the steps in the procedure in Dropping Tables and Views, page 2-19.

Step 2 If the required MIB attributes are not polled in the PollDefinition section, modify the polling definitions.

Step 3 Include the formulas required as shown in Modifying the ProcessPollResult Section, page 2-21.

Step 4 Add the variables in the database table using the *Var* element in the <ProcessDBSummary> element (see Modifying the ProcessDBSummary Section, page 2-20).

Step 5 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 2-20).

Step 2 If formula is required add it in the ProcessPollResult section (see Modifying the ProcessPollResult Section, page 2-21.

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, we add formulas in the ProcessPollResult section (See Modifying the ProcessPollResult Section, page 2-21) and to the ProcessDBSummary section as shown as 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 MIB table name in the MIBLevel section from which the MIB attribute is to be polled.

For example:

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

Here, CISCO_PROCESS_MIB is the MIB table name.

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 2-20.

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 (Cisco Active Network Abstraction), you can set up cross-launch of Prime Performance Manager reports from Cisco ANA NetworkVision 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 Choose **Administrative** from the menu tree.

Step 2 Click the Prime Network tab.

The Prime Network Gateway page appears.

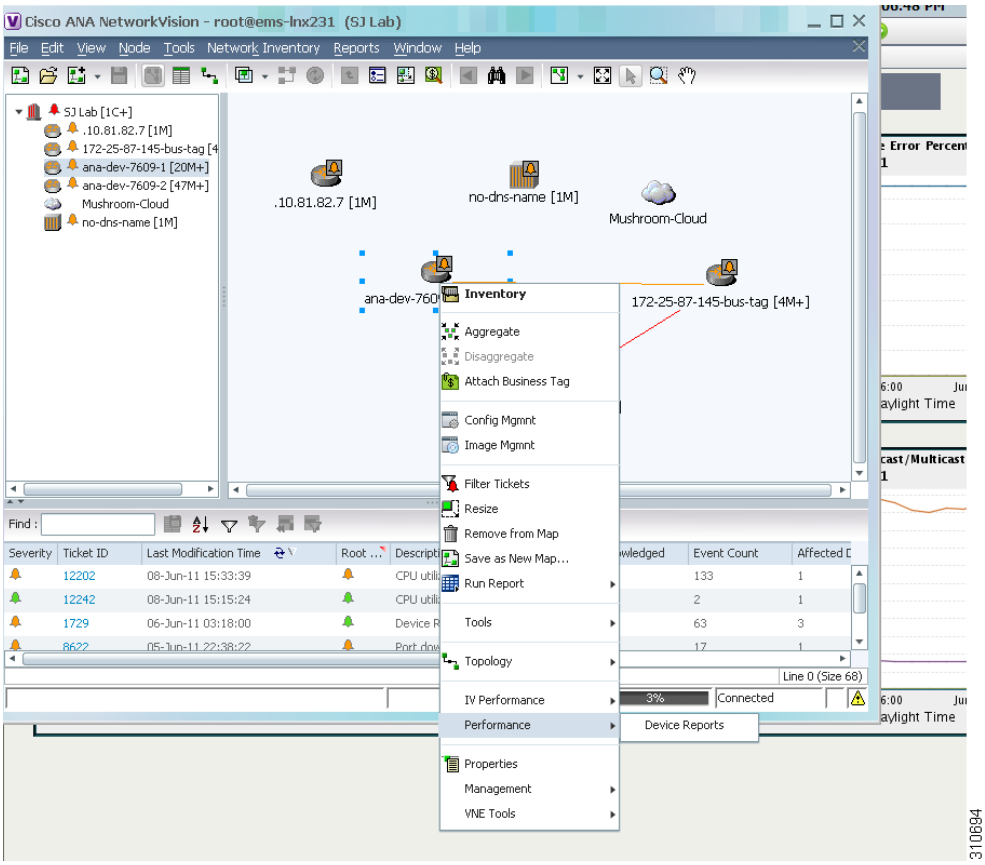
Step 3 On the Prime Network Gateway page, enter the parameters to log into the Prime Network Gateway:

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 ANA Device Shortcut menu.

ANA 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 ANA

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

Script Names	Description
bgp.bql, bgp-remove.bql	Adds, removes, cross-launch points for Border Gateway Protocol (BGP) reports.
evc.bql, evc-remove.bql	Adds, removes cross-launch points for Ethernet Virtual Circuit (EVC) reports.
if.bql, if-remove.bql	Adds, removes, cross-launch points for interface reports.
mpls-te-tunnel.bql, mpls-te-tunnel-remove.bql	Adds, removes, cross-launch points for MPLS Traffic Engineering reports.
node.bql, node-remove.bql	Adds, removes, cross-launch points for node level reports.
ospf.bql, ospf-remove.bql	Adds, removes, cross-launch points for OSPF reports.
pwe3.bql, pwe3-remove.bql	Adds, removes, cross-launch points for Pseudo1wire Emulation Edge-to-Edge (PWE3) reports.

Note that every BQL script to set 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 Cisco ANA.

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 Cisco ANA by modifying the provided BQL samples and then enabling cross launch.

CHAPTER 3

Testing and Debugging 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 check box next to your report and then click the **Save** icon.

The report should now be active.

Step 5 In the reports tree, click your report to start it.

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 ANA NetworkVision.
- **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 on **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 ANA are available and work correctly.

Common Issues and Error Messages

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

To see error messages:

Step 1 In the navigation tree, choose **Administrative**.

Step 2 Click the General tab.

Step 3 To see error messages, under the General heading, choose **Error Messages**.

Step 4 To see a console log, under System Logs, choose **Console Log**.

Database Tables Become Unstable

If your report modifies the same tables used by other reports, the database might become unstable.

Also, when there is a change to the existing database like adding a column or renaming a column or deleting a column, table and view drops are required. Make sure these drop views are done in both gateway and unit to avoid inconsistency.

For information on performing a table drop or view drop, see Dropping Tables and Views, page 2-19.

Incorrect MIB Variable

If your report uses an incorrect MIB variable, an error message will appear.

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

CHAPTER 4

XML Reference

This chapter provides reference information on the XML elements used in Prime Performance Manager reports.

Bits

Specifies a bit value. Used in the Column element.

Bytes

Specifies a byte value. Used in the Column element.

Column

The Column element is contained in a CSV 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 *SystemCapbility.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
- Util

- Bits
- Bytes
- Util
- IpAddr
- Time

GraphView

Specifies the graphs and subgraphs used in the Graph view for a report. Contains the following elements:

- 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 encapsulates the extension base "Processor," which specifies the following key elements of the report:

- 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:

- Label
- Link
- Util

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. The WebReport section contains:

- GraphView
- TableView

CHAPTER 5

Reports Macro Reference

The Cisco 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 the 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

BOOLEANVALUE

Syntax

BOOLEANVALUE (*object*)

Macro Description

Converts the object to Boolean form. Returns Null, if there is a failure.

BREAK

Syntax

BREAK()

Macro Description

- Used only in the ProcessPollResult section.
- Halts processing for that row and all subsequent rows.

CONTINUE

Syntax

CONTINUE()

Macro Description

- Used only in the ProcessPollResult section.
- Skips processing for that row and continues onto the next row.

DELTA

Syntax

DELTA (*object*)

Macro Description

Used only in the ProcessPollResult section.

The *object* parameter:

- Is a numeric type that returns the delta between the current and the previous poll value (currentPolling – previousPolling).
- Takes care of conditions that can occur such as when the first poll occurs (no previous value) and when the number overflows.

DELTA NEXT

Syntax

DELTANEXT *(object*, *arg1*)

Macro Description

- *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).
- Allows you to call the DELTA macro between the next row in the database instead of the previous polling.

This macro continues to calculate the delta until the row's value for *arg1* has changed.

DEVICETYPE

Syntax

DEVICETYPE (*object*)

Macro Description

- Used only in the *SystemCapabilities.xml* file, and should not be called again by a user.
- Returns the device type of the node (typically, *object* is sysObjectOID).

DOUBLEVALUE

Syntax

DOUBLEVALUE (*object*)

Macro Description

Converts the object (number or string) into a double and returns it. Returns Null, if there is a failure.

ENDSWITH

Syntax

ENDSWITH (*object*, *arg1*)

Macro Description

- *Object* is a string type and *arg1* is a string.
- Returns true if the string object ends with the string *arg1*.
- Similar to Java's endsWith string function.

FILTER

Syntax

FILTER (*object***,** *arg1*)

Macro Description

- *Object* is a table and *arg1* is conditions (returns a Boolean result).
- Returns a subset of objects with items that do not pass the conditions (items that return false) removed.

GETALL

Syntax

GETALL (*object*, *arg1*)

Macro Description

- *Object* is a table and arg1 is a string (that is a key).
- Returns all the column values of *arg1*.

GETAVAILABILITYINFO

Syntax

GETAVAILABILITYINFO()

Macro Description

- Used only in the PollDefinition section.
- 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.”

GETCACHE

Syntax

GETCACHE (*object*)

Macro Description

- *Object* is a key in the cache.
- Returns the value for the given key.
- The cache is a list of key/value pairs that makes the information viewable in other Poll Definitions later.

GETHOSTADDRESS

Syntax

GETHOSTADDRESS (*object*)

Macro Description

- Used only in ProcessPollResult section.
- *Object* is an IP address.
- Returns the host address in string form.

GETHOSTNAME

Syntax

GETHOSTNAME (*object*)

Macro Description

- Used only in the ProcessPollResult section.
- *Object* is an IP address.
- Returns the host name in string form.

HASCAPABILITY

Syntax

HASCAPABILITY(*arg1*)

Macro Description

Checks whether the device has capability for the MIB (given as a string as *arg1*).

HASVAR

Syntax

HASVAR (*arg1*)

Macro Description

- Used only in the *SystemCapability.xml* file.
- Returns True if the device has the given variable (*arg1*).

IF

Syntax

IF (**object**, *arg1*, [*arg2*])

Macro Description

Object is a Boolean type

- 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.

IFDESCR

Syntax

IFDESCR ([**object**])

Macro Description

- Used only in the ProcessPollResult section.
- Returns a description of the interface by looking it up first, to prevent re-polling.
- The optional argument is the index key for the operation (default is ifIndex).

IFSPEED

Syntax

IFSPEED ([*object*])

Macro Description

- Used only in the ProcessPollResult section.
- Returns the configured speed of the interface.
- The optional argument is the index key for the operation (default is `ifIndex`).

IFSPEEDRECEIVE

Syntax

IFSPEEDRECEIVE ([*object*])

Macro Description

- Returns the receive interface speed.
- The optional argument is the index key for the operation (default is *ifIndex*).

INDEXOF

Syntax

INDEXOF (*object*, *arg1*, [*arg2*])

Macro Description

- *Object* is a string type, *arg1* is a string, and *arg2* is an integer.
- If two arguments (*object* and *arg1*) are used, returns the starting index of the first occurrence of *arg1* in *object*.
- If three arguments are used, returns the starting index of the first occurrence of *arg1* in object located after index number *arg2*.
- Similar to Java's indexOf string function.

INTERVALDURATION

Syntax

INTERVALDURATION ()

Macro Description

- Used only in the WebReport or CSV section.
- 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.

INTVALUE

Syntax

INTVALUE (*object*)

Macro Description

Converts the object into an integer. Returns Null, if there is an error.

IOSVERSION

Syntax

IOSVERSION (*object*)

Macro Description

- Used only in the *SystemCapability.xml* file (only needs to be used once).
- If *object* is "sysDescr" (which is typical), will parse the IOS version from the given string.

IPADDRESS

Syntax

IPADDRESS (*object*, [*arg1*, *arg2*])

Macro Description

- Used only in the ProcessPollResult section.
- *Object* is an octet string address and *arg1*/*arg2* are offsets.
- 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.

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.

ISTABLEEMPTY

Syntax

ISTABLEEMPTY (*object*)

Macro Description

- Used only in the *SystemCapability.xml* file or the PollDefinition section.
- Object is a table.
- Returns True if object is empty.

JOIN

Syntax

JOIN (*object*, *arg1*, *arg2*)

Macro Description

- Used only in the PollDefinition section.
- *Object* and *arg1* are tables and *arg2* specifies a match condition.
- 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.

LASTVALUE

Syntax

LASTVALUE (*object*)

Macro Description

- Used only in the ProcessPollResult section.
- *Object* is a name of variable to retrieve.
- Returns the last value (or null if used for the first time).
- Used with the PERSISTVALUE macro.

LEFTJOIN

Syntax

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

Macro Description

- *Object* and *arg1* are tables and *arg2* is a match condition.
- 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*.

LENGTH

Syntax

LENGTH (*object*)

Macro Description

- *Object* is a string type.
- Returns the number of characters in object.
- Similar to Java's length string function.

LONGVALUE

Syntax

LONGVALUE (*object*)

Macro Description

Converts object into a long. Returns Null, if there is an error.

MATCHES

Syntax

MATCHES (*object*, *arg1*)

Macro Description

- *Object* is a string and *arg1* is a Java regular expression pattern.
- Returns True if the regular expression pattern *arg1* is found in *object*.

NEXTVALUE

Syntax

NEXTVALUE (*object*)

Macro Description

- Used only in the ProcessPollResult section.
- *Object* is a variable name.
- Returns the variable in the next table row, relative to the current row.

NOT

Syntax

NOT (*object*)

Macro Description

Object is a Boolean type.

Returns the opposite of *object*.

PERSISTVALUE

Syntax

PERSISTVALUE *(object*)

Macro Description

- *Object* is the name of the variable to save.
- Saves the current value of object across polls.
- Used with the LASTVALUE macro.

POLL

Syntax

POLL (*arg1*, *arg2*)

Macro Description

- Used only in the PollDefinition section.
- *Arg1* is a list of variables (comma delimited list of variables in double quotes) to poll and *arg2* is the label used to allow you to reference it in other polls to prevent re-polling.
- Polls only group of scalar values or table variables that share the same index.

Note that you need to poll all variables referenced in current and other reports with the same label in the same poll macro. This is to prevent re-polling.

POLLNEXT

Syntax

POLLNEXT (*arg1*, *arg2*, *arg3*, *arg4*)

Macro Description

- Used only in the PollDefinition section.
- The descriptions of the arguments are:
  - *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.
- Retrieves the data that you have not already retrieved (retrieves the next available data).

POLLPERSIST

Syntax

POLLPERSIST (*arg1*)

Macro Description

- Used only in the *SystemCapability.xml* file.
- *Arg1* is a list of variables to poll.
- Polls the values and puts it into the current context (available for immediate use in other capability checks).

PRINT

Syntax

PRINT ([*object*])

Macro Description

- 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).

PUTCACHE

Syntax

PUTCACHE (*object*, *arg1*)

Macro Description

- Object is a key that you want and *arg1* is a value that you want.
- Puts the key/value pair *object*/*arg1* into the cache.
- The cache is a list of key/value pairs that makes the information viewable in other Poll Definitions later.

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, *arg2* is the value used for time and *arg3* 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.

RETURN

Syntax

RETURN ()

Macro Description

- Used only in the PollDefinition section.
- Stops the execution of the current polling declaration.

SETCPUINFO

Syntax

SETCPUINFO ([*object*])

Macro Description

- Sets information about the CPU (*cpuDescr*, *cpuNum*, *cpuSlot*).
- To override the CPU index, set object to that value.

SETTIMEVARINFO

Syntax

SETTIMEVARINFO (*object*, *arg1*, *arg2*)

Macro Description

- *Object* is the time variable to use, *arg1* is a Boolean whether to use the variable next time, and *arg2* is the index in which you use until it changes value.
- Sets the time variable information with the given arguments in the context.

SHORTVALUE

Syntax

SHORTVALUE (*object*)

Macro Description

Returns the object as a short. Returns Null, if there is an error.

STARTSWITH

Syntax

STARTSWITH (*object*, *arg1*)

Macro Description

- *Object* is a string type and *arg1* is a string.
- Returns true if the string object starts with the string *arg1*.
- Similar to Java's startsWith string function.

SUBSTRING

Syntax

SUBSTRING (*object*, *arg1*, [*arg2*])

Macro Description

- *Object* is a string type and *arg1*/arg2 *are* integers.
- If two arguments are used, returns a string that starts at the *arg1* index of *object.*
- If three arguments are used, returns a string that starts at the *arg1* index of object and ends at the *arg2* index of *object*.
- If the last argument is larger than the size of *object*, returns null.
- Similar to Java's substring string function.

SYSTIME

Syntax

SYSTIME ()

Macro Description

Returns the current system time (in milliseconds).

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).

TOPN

Syntax

TOPN (*object*,*arg1*,*arg2*)

Macro Description

- Used in the PollDefinition or ProcessDBSummary section.
- *Object* is a table, *arg1* is what column to sort by, and *arg2* is the number of records to get (*n*).
- Returns the top *n* (*arg2*) rows sorted by a sort key descending.
- 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.

TOSTRING

Syntax

TOSTRING (*object*)

Macro Description

Returns *object* in string form.

INDEX

A

B

D

F

I

S

V