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"))
```