CodeMagus

cmlxwinp: Windows Performance Metric Probe

CML00048-01

Code Magus Limited (England reg. no. 4024745)
Number 6, 69 Woodstock Road
Oxford, OX2 6EY, United Kingdom
www.codemagus.com

CodeMagus

December 15, 2020

Contents

1 Introduction

This document describes how to use `cmlxwinp` which is a software utility to fetch WINDOWS performance counters using the Performance Data Helper (PDH) application programming interface. The data is fed to a `Serfboard` server for use in displaying a real time dashboard and stored for post processing analysis.

In order to process metrics from the raw WINDOWS platform and send them to `Serfboard` in the correct form `cmlxwinp` needs to be configured. This is done through its command interface.

The Performance Data Helper application programming interface (API) is a collection of C programming language subroutines that execute in user space. This API is used to collect performance data of various performance counters or instances that are available on the system.

This application reads the values of a list of performance counters to be fetched from the local WINDOWS host. Each metric is then formatted as a `Serfboard` metric and sent to a `Serfboard` server.

For `Serfboard` documentation please refer to the following manuals:

- Serfboard Configuration Guide and Reference Version 1 [3]
- Serfboard Instruments Guide and Reference Version 1 [4]
- Serfboard User Guide Version 1 [5]

2 Synopsis

`cmlxwinp` is invoked from the command line and if the '`--help`' parameter is specified it will display all available parameters and their options if applicable. Below is the help display and following that is a description of each parameter.

```
Code Magus Limited cmlxwinp V1.0: build 2010-12-01-13.12.58
[cmlxwinp] $Id: cmlxwinp.c,v 1.7 2010/11/22 13:18:09 janvlok Exp $
Copyright (c) 2009 by Code Magus Limited. All rights reserved.
 [Contact: stephen@codemagus.com].
Usage: cmlxwinp [OPTION...]
  -p, --port={60051|<port>}   Command interface port
  -c, --command=<command>     Command to pass to command process
  -v, --verbose               Verbose output
  -t, --trace                 Trace message output

Help options
  -?, --help                  Show this help message
  --usage                     Display brief usage message
```

Where:

- '`-p|--port`' Specifies the command interface port for `cmlxwinp`, If not specified it will default to 60051.
- '`-c|--command`' Specifies a command to be passed to the command interface.
- '`-v|--verbose`' When specified, `cmlxwinp` operates in a verbose manner.
- '`-t|--trace`' When specified, `cmlxwinp` writes all activity to `stdout`.

3 Invocation

When `cmlxwinp` is invoked it starts a command interface through which the processing of the probe is configured and listens on a TCP/IP port (see invocation parameters in sub-section 2 on page 3) for connections. The command supplied as a parameter is presented directly to the command interface, followed by commands supplied through connections to the TCP/IP port, either interactively with a client like `telnet` or through `cmlcmd` [2].

Once configured the metrics and definitions may also be viewed or the `cmlxwinp` environment dynamically reconfigured via further commands.

The following sub-sections describe the `cmlxwinp` command interface and commands.

4 Command Interface

`cmlxwinp` is configured from commands presented to its command interface. Input is either a single command or the name of a command file. Commands are explained in detail in the following sections. A command file is a text based file that consists of one or more commands, where each command is on a separate line. Typically a probe requires multiple commands to be effectively configured so commands are often written as a logical group in a command file. A command file name is validated using the library cmdname [6].

5 Command Elements

5.1 Comments

Comments are introduced by using a double minus ('`--`') and continue up to the end of the current input line.

Examples:

```
-- This is an example of a command comment.
-- and is useful in documenting command files.
```

5.2 Reserved Words

Reserved words have a special meaning in terms of directing the parsing of commands. The reserved words are:

```
add         group      parameter    udp
all         help       parameters   trace
available   inactive   probe        verbose
close       interval   server
counter     log        set
counters    metric     shutdown
delete      msglevel   start
delay       notset     stop
display     open       switch
exit        polling    title
```

5.3 Identifiers

Identifiers are case sensitive and start with a letter which can be followed by any number of letters, digits, decimal point '.' or the under-score character.

Examples:

```
cpu_total_processor memory_committed_bytes
```

5.4 Strings

Strings are:

- any sequence of characters (except double quotes and the newline character) enclosed by double quotes.
- any sequence of characters (except single quotes and the newline character) enclosed by single quotes.

Examples:

```
"Seconds spent in user mode"
"ABC Company's Metric File"
'$Revision: 1.13 $'
```

5.5 Integers

An integer consists of a nonempty sequence of decimal digits.

Examples:

```
1234
0
```

5.6 Numbers

A number consists of a nonempty sequence of decimal digits that

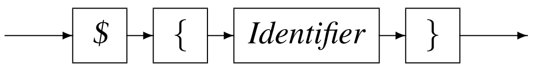
- possibly contains a radix character (decimal point '.').
- is optionally followed by a decimal exponent; consisting of an 'E' or 'e' followed by an optional plus or minus sign followed by a nonempty sequence of decimal digits that indicates multiplication by a power of 10.

Examples:

```
1234
0.001
1.2
123.45E-12
```

5.7 Environment Variables

EnvironmentVariable



Environment variables are substituted by their value when encountered in command input text.

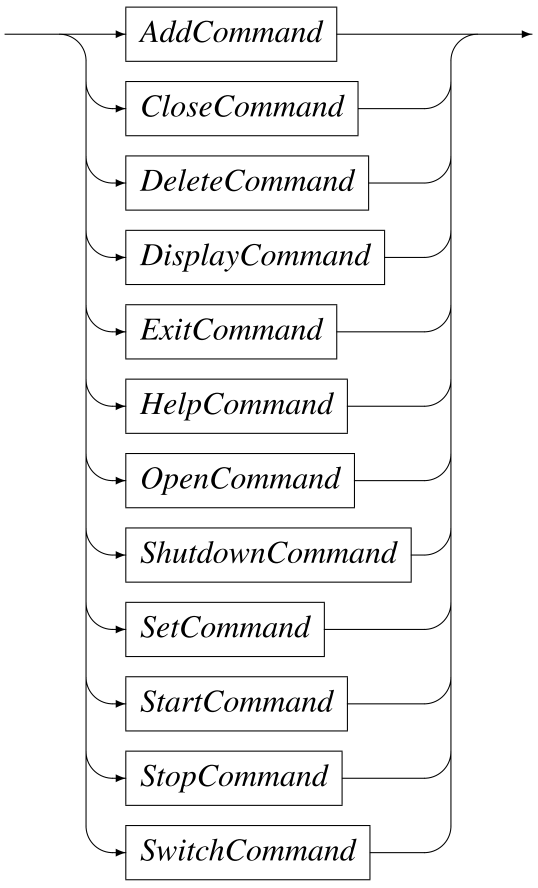
6 Syntax and Semantics

Input to the command processor is either:

- A *Comment*. The whole line is ignored by the command processor, see subsection 5.1 on page 4.
- A *Command*.
- A *Command File Name*. If the input is not a command, the command processor interprets the input as a command file name and, after validating it with `cmdname` [6], will attempt to open it and process each command within it.

See appendix A on page 18 for an example of a `cmlxwinp` command file.

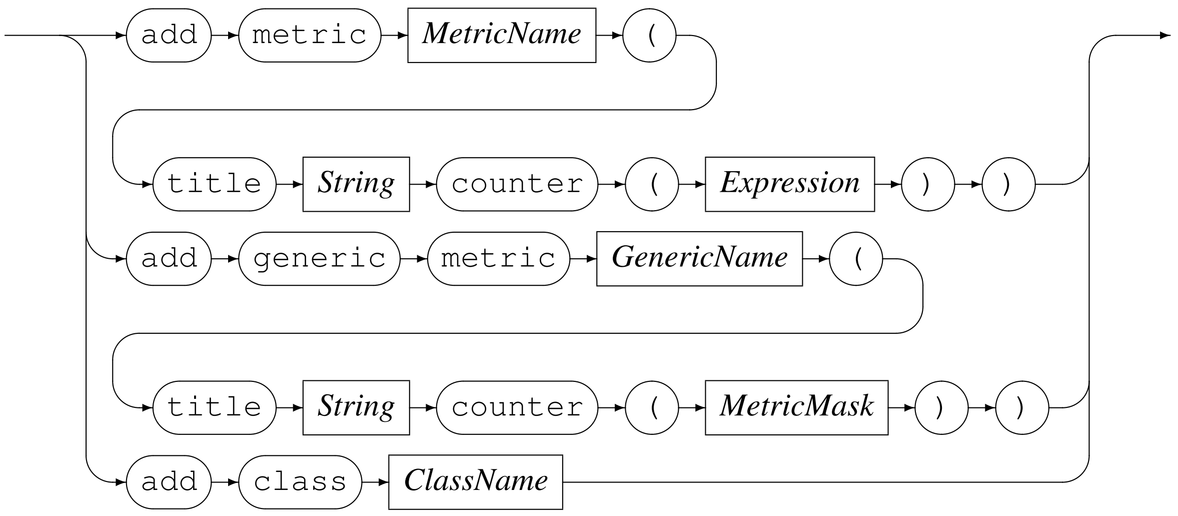
command



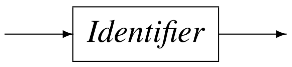
6.1 Add Command

This command is used to add a definition of a metric to be extracted. It describes the binding of the WINDOWS performance counters to a `Serfboard` metric.

AddCommand



MetricName



In the first form *MetricName* is the name of the configured metric in the `Serfboard` server. *Expression* is a regular arithmetic expression where the variables are WINDOWS performance counters.

In the second form *GenericName* is the name of a generic metric where the associated *MetricMask* is a regular expression that matches one or more metrics configured within the `Serfboard` server.

In the third form *ClassName* is the name of the configured class of metrics associated with the `cmlxwinp` probe. The classes provided are:

- cpu - This includes the following metrics:
  - Total Handle Count
  - Percent Processor Time
  - Percent User Time
  - Percent Privileged Time
  - Percent Idle Time
- memory - This includes the following metrics:
  - Memory Available Bytes
  - Memory Committed Bytes
- disk - This includes the following metrics:

  - Disk reads per second
  - Disk writes per second
  - Disk bytes read per second
  - Disk byres write per second
- netinterface - This includes the following metrics:
  - Number of packets received per second
  - Number of packets sent per second
  - Number of bytes received per second
  - Number of bytes sent per second

Example

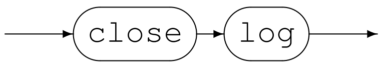
The `Serfboard` metric `cpu_total` is the summation of the two WINDOWS performance counters (“`\Processor(0)\% User Time`” and “`\Processor(0)\% Privileged Time`”):

```
WIN example> add metric cpu_total (\
   title "Privileged plus User"\
   counter ("\Processor(0)\% User Time" +\
            "\Processor(_Total)\% Privileged Time")\
   )
Added metric cpu_total
```

6.2 Close Command

This command is used to either:

- Close an opened recording log:

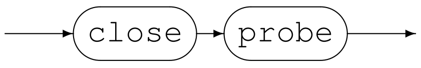


Example:

Close the previous opened log:

```
WIN example> close log
Log "text(example_probe.txt,mode=w)" closed, record count = 11
```

- Close the active probe:



Example:

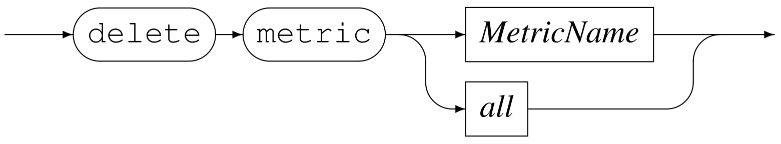
Close an opened probe:

```
WIN example> close probe
WIN example>
```

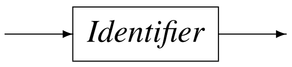
6.3 Delete Command

This command will delete the definition of a previously defined metric and can not be performed when a probe has been started. The metric data will no longer be extracted and sent to a `Serfboard` server when the probe is restarted.

DeleteCommand



MetricName



Example:

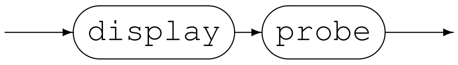
Delete `Serfboard` metric `disk_tot_nread`:

```
WIN example> delete metric disk_tot_nread
Error: Probe is active - metric maintenance suspended!
WIN example> stop probe
Probing stopped
WIN example> delete metric disk_tot_nread
Metric disk_tot_nread deleted
```

6.4 Display Command

This command is used to display various configuration settings:

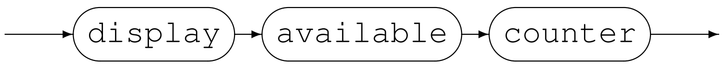
- Probe status:



Example:

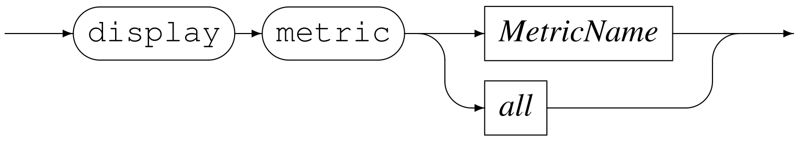
```
WIN example> display probe
Probe cmlxwinp
   Status:          Active
   Title:           "WIN example"
   Group:            example
   Parameters:       ''
   Polling Interval: 60
   Inactive delay:   36000
   Server:    codemagus.it.nednet.co.za:41059 UDP Connected
```

- Available WINDOWS performance counters for probing:

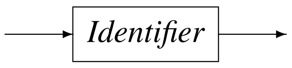


```
WIN example> display available counters
\SMSvcHost 3.0.0.0\Protocol Failures over net.tcp
\SMSvcHost 3.0.0.0\Protocol Failures over net.pipe
\SMSvcHost 3.0.0.0\Dispatch Failures over net.tcp
.
.
.
```

- Display metrics:



MetricName

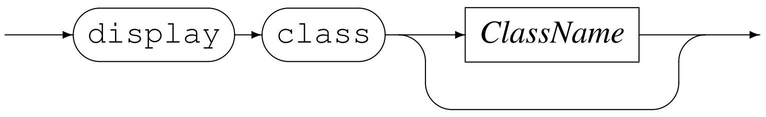


Example:

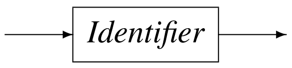
Display all the metrics that have been defined:

```
WIN example> display metric all
WIN example> display metric all
metric cpu_total_processor
   (
   title "Active Clock seconds"
   counter ("\Processor(_Total)\% Processor Time")   )
   )
metric cpu_total_user
.
.
.
```

- Display classes:

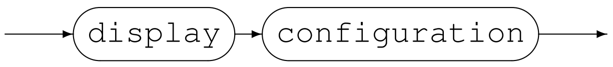


ClassName



Display the class defined by *ClassName* or all classes if it is omitted.

- Display configuration:

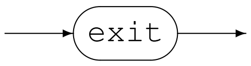


Display the current configuration in effect.

6.5 Exit Command

This command terminates an interactive session to the command interface of `cmlxwinp` and disconnects the client from the TCP/IP port.

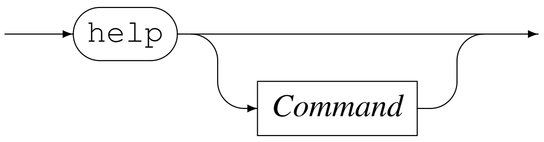
ExitCommand



6.6 Help Command

Help on `cmlxwinp` commands.

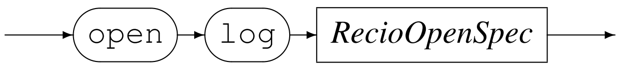
HelpCommand



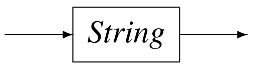
6.7 Open Command

This command is used to either

- Open a log file for recording the metrics sent to `Serfboard`.



RecioOpenSpec



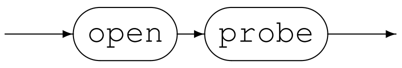
RecioOpenSpec is a `recio` [1] open specification string.

Example:

Open a log file for recording the metrics. The two environment variables will be expanded to the current date and time respectively.

```
WIN example> open log \
           "text(example_probe_D${DATE_YYMMDD}_T${TIME_HHMMSS}.txt.txt,mode=w)"
Log "text(example_probe_D101210_T103857.txt.txt,mode=w)" opened
```

- Initialise the probe:



Example:

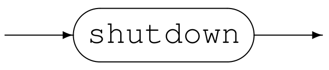
Initialise the probe by opening it:

```
WIN example> open probe
CML WIN probe initialised
```

6.8 Shutdown Command

This command terminates `cmlxwinp`.

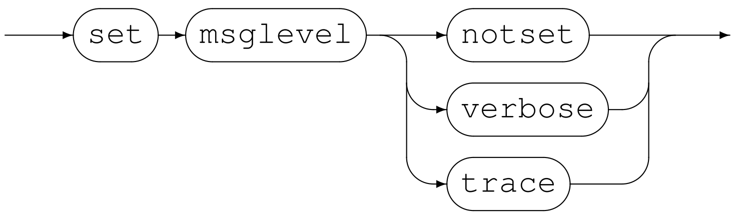
ShutdownCommand



6.9 Set Command

This command is use to set the internal variables and parameters of `cmlxwinp`. If the `set` command results in changing an internal variable or a parameter the response to the `set` command is to display the probe status. The following can be set:

- Set the level of diagnostics produced by `cmlxwinp`:

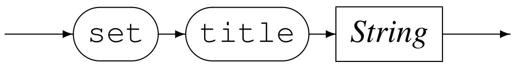


Example:

Reset the level to not produce diagnostic messages:

```
WIN example> set msglevel notset
WIN example>
```

- Set the title for `cmlxwinp`, this is used for the prompt of the command interface:

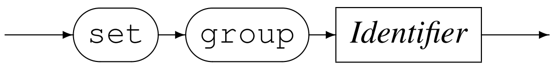


Example:

Set the title of the probe to 'WIN example':

```
WIN performance counters> set title "WIN example"
Probe cmlxwinp
   Status:            Initialised
   Title:             "WIN example"
   Group:             cmlxwinp
   Parameters:        ''
   Polling Interval: 30
   Inactive delay:   18000
   Server:    Not specified
WIN example>
```

- Set the `Serfboard` group name:

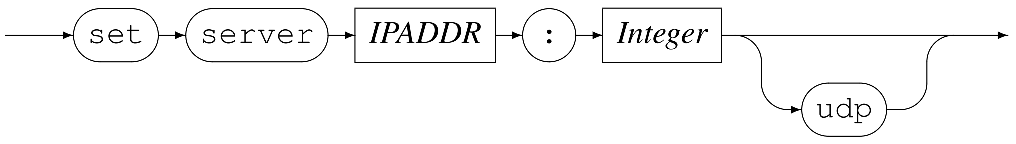


Example:

Set the `Serfboard` group name to `example`:

```
WIN example> set group example
Probe cmlxwinp
   Status:            Initialised
   Title:             "WIN example"
   Group:             example
   Parameters:        ''
   Polling Interval: 30
   Inactive delay:   18000
   Server:    Not specified
WIN example>
```

- Set the host address of the `Serfboard` server to which the metrics are sent:



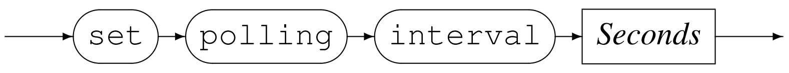
IPAddress can be specified as a host name or by using the Internet notation of dots and numbers. The default connection is TCP/IP, but if *udp* is specified, UDP will be used, a connectionless transport without guarantee of delivery.

Example:

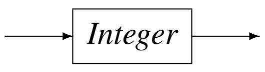
Set the host address of the `Serfboard` server to `codemagus.it.nednet.co.za`, listening on port `41000` and use UDP :

```
WIN example> set server codemagus.it.nednet.co.za:41000 UDP
Probe cmlxwinp
   Status:            Initialised
   Title:             "WIN example"
   Group:             example
   Parameters:        ''
   Polling Interval: 30
   Inactive delay:   18000
   Server:    codemagus.it.nednet.co.za:41000 UDP Not Connected
WIN example>
```

- Set the polling interval:



Seconds



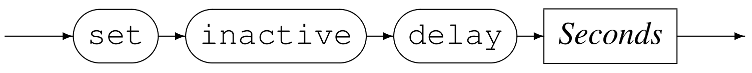
The polling interval is specified in seconds and is the frequency at which the WINDOWS performance counters will be polled in order to extract metric data. The frequency must be greater than zero and less than 100.

Example:

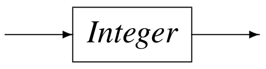
Set the polling interval to one minute:

```
WIN example> set polling interval 60
Probe cmlxwinp
   Status:            Initialised
   Title:             "WIN example"
   Group:             example
   Parameters:        ''
   Polling Interval: 60
   Inactive delay:   18000
   Server:    codemagus.it.nednet.co.za:41000 UDP Not Connected
WIN example>
```

- Set inactive delay:



Seconds



The inactive delay is specified in seconds and the default value is equivalent to five hours. The `DELAY` defines the amount of time from the last command processed by the command interface before the probe automatically stops extracting metric data by performing a '`stop probe`' command internally. This feature is always active and prevents the probe from flooding the network with metric data when no longer required.

Example:

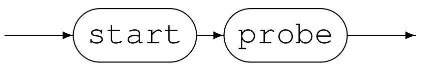
Set the inactive delay to one hour:

```
WIN example> set inactive delay 36000
Probe cmlxwinp
   Status:           Initialised
   Title:            "WIN example"
   Group:            example
   Parameters:       ''
   Polling Interval: 60
   Inactive delay:   36000
   Server:    codemagus.it.nednet.co.za:41000 UDP Not Connected
WIN example>
```

6.10 Start Command

This command causes the probe to start extracting metric data and sending it on to `Serfboard`.

StartCommand



Examples:

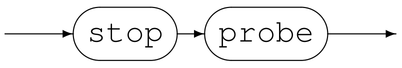
Start probing:

```
WIN example> start probe
Probing started
```

6.11 Stop Command

This command causes the probe to stop extracting metric data and sending it on to `Serfboard`; `cmlxwinp`() reverts back to the idle state.

StopCommand



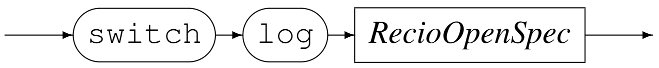
Examples:

Stop probing:

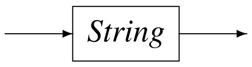
```
WIN example> stop probe
Probing stopped
```

6.12 Switch Command

Close a previous recording log file and open a new one.



RecioOpenSpec



RecioOpenSpec is a `recio` [1] open specification string.

Example:

Close the current log file and open a new log file for recording the metrics. The two environment variables will be expanded to the current date and time respectively.

```
WIN example> switch log \
           "text(example_probe_D${DATE_YYMMDD}_T${TIME_HHMMSS}.txt.txt,mode=w)"
Log "text(example_probe_D101210_T103857.txt.txt,mode=w)" closed,
     record count = 0
Log "text(example_probe_D101210_T105920.txt.txt,mode=w)" opened
```

A Example Command File

```
-- File: example.cmd
--
-- WIN PERF stats feed.
--
-- Copyright (c) 2010 Code Magus Limited. All rights reserved.
--
-- $Author: janvlok $
-- $Date: 2011/01/03 10:14:21 $
-- $Id: example.cmd,v 1.1 2011/01/03 10:14:21 janvlok Exp $
-- $Source: /home/cvs/cvsroot/cmlxwinp/documents/example.cmd,v $
-- $Revision: 1.1 $
-- $State: Exp $
--
-- $Log: example.cmd,v $
-- Revision 1.1  2011/01/03 10:14:21  janvlok
-- Re-write of the probe
--
-- first time we need the select, later we need the close
close probe
set title "WIN example"
set group example
set server codemagus.it.nednet.co.za:41059 UDP
set polling interval 60
--
delete metric all
-- CPU
add metric cpu_total_processor (\
   title "Active Clock seconds"\
   counter ("\Processor(_Total)\% Processor Time")\
   )
add metric cpu_total_user (\
   title "Clock seconds spent in user mode"\
   counter ("\Processor(0)\% User Time")\
   )
add metric cpu_total_privileged (\
   title "Clock seconds spent in privileged mode"\
   counter ("\Processor(_Total)\% Privileged Time")\
   )
add metric cpu_total_idle (\
   title "Clock ticks spent in idle mode"\
   counter ("\Processor(_Total)\% Idle Time")\
   )
add metric cpu_total (\
   title "Privileged plus User"\
   counter ("\Processor(0)\% User Time" +\
      "\Processor(_Total)\% Privileged Time")\
   )
-- Memory
add metric memory_available_bytes (\
   title "Memory Available Bytes"\
   counter ("\Memory\Available Bytes")\
   )
add metric memory_committed_bytes (\
   title "Memory Committed Bytes"\
   counter ("\Memory\Committed Bytes")\
   )
-- Disk
add metric disk_reads (\
   title "Disk Reads"\
   counter ("\PhysicalDisk(_Total)\Disk Reads/sec")\
   )
add metric disk_writes (\
   title "Disk Writes"\
   counter ("\PhysicalDisk(_Total)\Disk Writes/sec")\
   )
add metric disk_bytes_read (\
   title "Disk Bytes Read"\
   counter ("\PhysicalDisk(_Total)\Disk Read Bytes/sec")\
   )
add metric disk_bytes_write (\
   title "Disk Bytes Write"\
   counter ("\PhysicalDisk(_Total)\Disk Write Bytes/sec")\
   )
-- Net interface
add metric net_ibytes (\
   title "Number of bytes received"\
   counter (\
   "\Network Interface(AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Minip
   )
add metric net_obytes (\
   title "Number of bytes sent"\
   counter (\
   "\Network Interface(AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Minip
   )
add metric net_ipackets (\
   title "Number of packets received"\
   counter (\
   "\Network Interface(AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Minip
   )
add metric net_opackets (\
   title "Number of packets sent"\
   counter (\
   "\Network Interface(AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Minip
   )
display metric all
open probe
start probe
```

References

[1] recio: Record Stream I/O Library Version 1. CML Document CML00001-01, Code Magus Limited, July 2008. PDF.

[2] cmlcmd: Command Utility Version 1. CML Document CML00007-01, Code Magus Limited, July 2008. PDF.

[3] Serfboard Configuration Guide and Reference Version 1. CML Document CML00023-01, Code Magus Limited, July 2008. PDF.

[4] Serfboard Instruments Guide and Reference Version 1. CML Document CML00024-01, Code Magus Limited, July 2008. PDF.

[5] Serfboard User Guide Version 1. CML Document CML00027-01, Code Magus Limited, July 2008. PDF.

[6] cmdname: Command Name Resolver Library Version 1. CML Document CML00076-01, Code Magus Limited, December 2010. PDF.