

HP GlancePlus

for the Linux operating system

Software Version: 11.00

User's Guide

Document Release Date: October 2010

Software Release Date: October 2010



Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notices

© Copyright 2010 Hewlett-Packard Development Company, L.P.

Trademark Notices

Adobe®, Acrobat®, and PostScript® are trademarks of Adobe Systems Incorporated.

Intel®, Itanium®, and Pentium® are trademarks of Intel Corporation in the U.S. and other countries.

Microsoft®, Windows®, Windows® XP, and Windows Vista® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Acknowledgements

This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes software written by Tim Hudson (tjh@cryptsoft.com).

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to:

<http://h20230.www2.hp.com/selfsolve/manuals>

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support web site at:

www.hp.com/go/hpsoftwaresupport

This Web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract. To register for an HP Passport ID, go to:

<http://h20229.www2.hp.com/passport-registration.html>

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

Contents

WARRANTY.....	2
RESTRICTED RIGHTS LEGEND.....	2
TRADEMARK NOTICES.....	2
CONTENTS.....	5
ABOUT GLANCEPLUS	21
GLANCEPLUS CONCEPTS	21
MANAGING SYSTEM RESOURCES.....	21

CPU INFORMATION	22
MEMORY INFORMATION	23
DISK INFORMATION	24
NETWORK INFORMATION	25
SYSTEM INFORMATION.....	26
MONITORING YOUR SYSTEM WITH THE ADVISER	31
CONFIGURE COLORS	33
CONFIGURE GRAPH LIMITS.....	37

CONFIGURE PROCESS FILTERS	37
GUIDED TOUR.....	38
GUIDED TOUR - GLANCEPLUS MAIN WINDOW	39
GUIDED TOUR - CPU BOTTLENECK, PANEL 1.....	39
GUIDED TOUR - CPU BOTTLENECK, PANEL 2.....	40
GUIDED TOUR - MEMORY BOTTLENECK, PANEL 1	41
GUIDED TOUR - MEMORY BOTTLENECK, PANEL 2	42
GUIDED TOUR - CONFIGURING GLANCEPLUS.....	43

GUIDED TOUR - ALARMS & SYMPTOMS	44
CUSTOMIZE GLANCEPLUS START-UP	45
OVERRIDE CONFIGURATION UPDATES.....	46
CHANGE DISPLAY COLORS.....	50
SORT REPORT FIELDS	53
SET FILTERS.....	54
HIGHLIGHT METRICS	55
MODIFYING RUNTIME PARAMETERS	56

FONT TYPE AND SIZE	56
ICON SIZE	56
COLUMN WIDTH	57
RESOURCES TABLE	57
VERSION WINDOW DISPLAY TIME	57
TRUNCATION RULES	58
DISPLAY TRUNCATED STRING DATA.....	58
SPEED KEYS.....	58

CMD.....	63
JAVAARG	63
ARGV1	63
INTRODUCTION TO THE GLANCEPLUS ADVISER.....	64
ALARMS AND SYMPTOMS	64
WHAT IS AN ALARM?	64
WHAT IS A SYMPTOM?	65
EDITING ADVISER SYNTAX.....	65

USING THE GLANCEPLUS TEXT EDITOR	65
USING YOUR OWN TEXT EDITOR.....	66
PRINTING CPU UTILIZATION DURING HIGH CPU USAGE.....	67
SENDING EMAIL MESSAGES	68
PRINTING PROCESS INFORMATION WITHIN A LOOP.....	68
PRINT TO A FILE.....	68
ADVISER SYNTAX STRUCTURE	69
ADVISER SYNTAX REFERENCE	70

SYNTAX CONVENTIONS	70
COMMENTS	71
CONDITION EXAMPLES	71
CONSTANTS	71
EXPRESSIONS	71
PRINTLIST	71
PRINTLIST EXAMPLES.....	72
METRIC NAMES IN ADVISER SYNTAX	72

VARIABLES	73
ALARM STATEMENT	73
ALARM EXAMPLES	73
ALARM EXAMPLE: PROCESS TABLE	74
ALARM EXAMPLE: SWAP SPACE	74
ALARM EXAMPLE: YELLOW ALERT	75
ALARM EXAMPLE: CPU PROBLEM	75
ALERT STATEMENT	75

ALIAS STATEMENT	76
ASSIGNMENT STATEMENT	76
COMPOUND STATEMENT.....	76
EXEC STATEMENT	77
GPM STATEMENT	77
IF STATEMENT.....	77
LOOP STATEMENT	78
TT LOOP EXAMPLE	82

TTBIN LOOP EXAMPLE	82
PRINT STATEMENT	85
SYMPTOM STATEMENT.....	85
SYMPTOM EXAMPLE: GLOBAL CPU BOTTLENECK	86
INTERVAL.....	87
GLANCEPLUS MESSAGES.....	87
ADVISER SYNTAX MESSAGES	87
GENERAL MESSAGES (AS-101 THROUGH AS-131)	87

ALARM MESSAGES (AS-201 THROUGH AS-212)	90
SYMPTOM MESSAGES (AS-301 THROUGH AS-306)	91
STATEMENT MESSAGES (AS-401 THROUGH AS-410)	91
ACTION MESSAGES (AS-501 THROUGH AS-504)	92
LOOP MESSAGES (AS-601 THROUGH AS-636)	92
MESSAGE AS-613	94
MESSAGE AS-614	94
MESSAGE AS-615	94

MESSAGE AS-618.....94

MESSAGE AS-619.....94

MESSAGE AS-621.....95

MESSAGE AS-622.....95

MESSAGE AS-623.....95

MESSAGE AS-624.....95

MESSAGE AS-625.....95

MESSAGE AS-626.....95

MESSAGE AS-627	95
MESSAGE AS-628	95
MESSAGE AS-629	95
MESSAGE AS-630	95
MESSAGE AS-633	95
MESSAGE AS-634	95
MESSAGE AS-635	95
TROUBLESHOOTING	96

INSTALLATION MESSAGES	96
START-UP MESSAGES	96
RUNNING IN BACKGROUND MODE.....	96
DURING INITIALIZATION START-UP	97
MESSAGES BEFORE CONNECTING TO DISPLAY	97
MESSAGES WHILE CONNECTING TO DISPLAY.....	97
MESSAGES AFTER CONNECTING TO DISPLAY	98
SUCCESSFUL INITIALIZATION.....	99

About GlancePlus

GlancePlus is a real-time performance monitoring and diagnostic tool that helps you get the best possible performance from your computer system. Real-time means that GlancePlus shows you, in words and pictures, exactly what's going on inside your computer right now. You define the time interval yourself – “right now” can mean what your computer was doing a second, a minute, or an hour ago.

Here are some of the features of GlancePlus:

- The rules-based adviser interprets performance data and identifies bottlenecks. You can use the default set of rules or create your own rules.
- Graphical and tabular on-screen color reports show your system's performance and resource utilization in real-time.
- You can view metrics at the global, application, or process level.
- You can use alarms to monitor your system. You can use the default alarm thresholds or define your own.
- You can define the order of the data columns and sort criteria. You can also define filters to see only information of interest to you.
- There are a variety of display options, colors, and fonts. You can define the data sample intervals and durations for graphs.
- You can enable/disable tooltips (click the “T” button in a row-column report window) to see the full text of truncated string data.
- You can monitor system performance while tending to other tasks by using GlancePlus in icon mode.
- The online help facility provides quick answers to your questions about GlancePlus.

GlancePlus Concepts

This section discusses some of the important concepts used in GlancePlus and explains how to manage system resources and processes, monitor applications and use the Adviser to monitor your system.

Because the illustrations in the Concepts section are large, you may want to enlarge your help window before you begin.

Click a highlighted topic for information about a specific concept.

[Managing System Resources](#)

[Monitoring Applications](#)

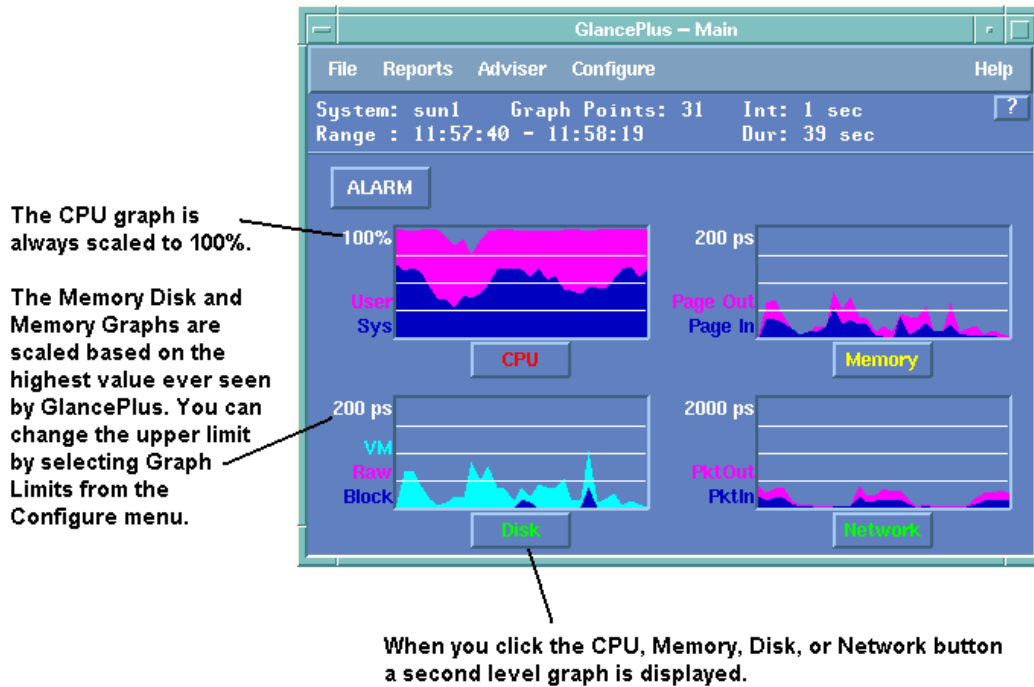
[Managing Processes](#)

[Monitoring Your System with the Adviser](#)

[Configuring GlancePlus](#)

Managing System Resources

You can display the information in the GlancePlus Main window as horizontal bars, vertical bars, pie charts, or as a resource history graph, which is shown here.



Select from the list below to display examples of graphs and reports.

[CPU Information](#)

[Disk Information](#)

[Memory Information](#)

[Network Information](#)

[System Information](#)

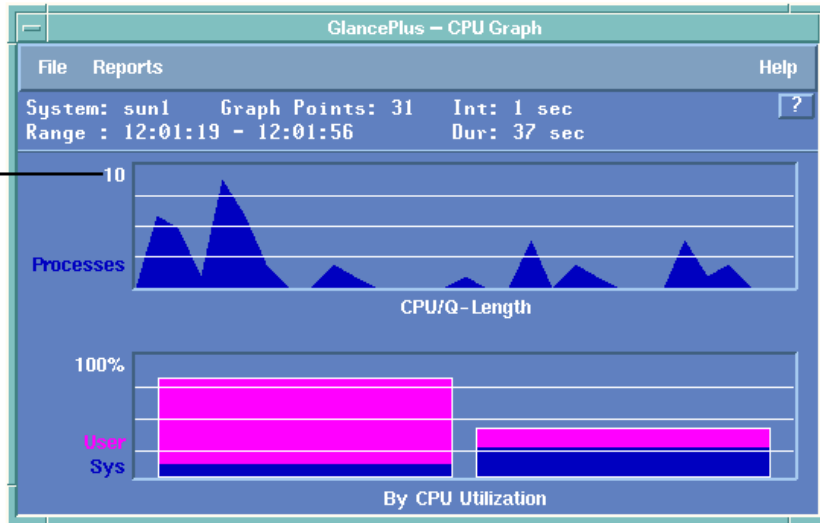
To select a graph style, see the Change Main Graph Style task.

CPU Information

You can use several windows, available from the Reports menu in the GlancePlus main window, to monitor CPU activity. You can also display the CPU graph by clicking the CPU button in the Main window.

The CPU graph is scaled based on the highest value found in the history buffer.

It depicts CPU queue length over time.



You can also display the CPU By Processor report.

To view the CPU By Processor report or the CPU graph, use the Reports menu.

The figure shows the 'GlancePlus - CPU Report' window. It contains a table with the following data:

State	Current	Avg.	High	Time	Cum Time
User	4.5%	24.4%	73.5%	0.1	204.4
System	13.4%	27.6%	100.0%	0.2	231.2
Idle	82.1%	48.0%	83.8%	0.9	402.0

Activity	Rate	C Rate	High
Syscalls	3216.3	5703.9	11004.5
Interrupts	473.6	1231.7	5975.0
Cont Switches	180.0	331.9	762.7

Queues	Length	Avg.	High
CPU Queue	0.0	1.3	9.0
Load Average	1.5	2.4	3.6

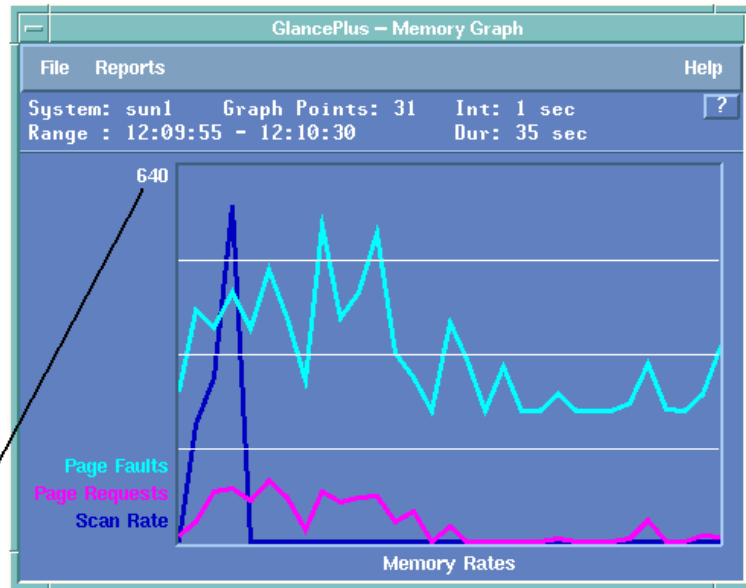
Memory Information

You can use several windows, available from the Reports menu in the GlancePlus Main window, to monitor memory activity. You can also display the Memory graph by clicking the Memory button in the Main window.

The Memory graph shows number of pages being freed per second. It also shows how many pages were scanned to find candidates for freeing per second.

This can be helpful in determining if there is memory pressure.

The graph's upper limit is based on the highest value found in the history buffer.



The Memory report displays tabular data showing the breakdown of memory available for certain operations.

The figure is a table titled "GlancePlus - Memory Report". It provides a detailed breakdown of memory-related statistics. The table has the following structure:

Event	Current	Cumulative	Curr Rate	Cum Rate	High Rate
Page Scans	0	391062	0.0	305.7	9450.0
Page Faults	472	441863	429.0	345.5	1551.8
Paging Requests	8	49774	7.2	38.9	232.7
Paged In (kb)	32	182624	29.0	142.7	1312.7
Paged Out (kb)	0	128624	0.0	100.5	1400.0

Below the table, additional memory statistics are listed:

- Phys Mem : 64.0mb
- Avail Mem: 53.1mb
- Free Mem : 1.0mb
- Buf Cache: 3.1mb

The title bar shows "System: sun1", "Last Update: 12:14:13", and "Int: 1 sec".

Disk Information

You can use several windows, available from the Reports menu in the GlancePlus Main window, to monitor disk resources. You can also display the Disk graph by clicking the Disk button on the Main window.

The Disk graph shows the top ten disks by utilization.

The IO By Disk report displays data for the active disks.

To get more detail on a particular disk, click that device name.

Device Name	Curr Queue	Req Queue	Disk %	Phys IO Rt
c0t3d0s*	0.0	0.4	0.0	0.9
c0t1d0s*	1.0	0.4	93.8	65.4
		0	0.0	0.0

The Disk report shows global information on the types and rate of disk activity.

Req Type	Requests	%	Rate	Bytes	Cum Req	%	Cum Rate	Cum Bytes
Logical Reads	0	0.0%	0.0	100189	78.4%	67.8		
Logical Writes	0	0.0%	0.0	27545	21.6%	18.6		
Read Sys Calls	159	37.0%	144.5	6kb				
Write Sys Calls	5	3.0%	4.5	7kb				
Phys Reads	0	0.0%	0.0	0kb				
Phys Writes	0	0.0%	0.0	0kb				
Virtual Mem	0	0.0%	0.0					
Raw	0	0.0%	0.0					
Block	0	0.0%	0.0					
NFS Server	0	0.0%	0.0					
NFS Client	0	0.0%	0.0					
Event	Current	Cumulative	Curr %					
Cache Hits	0	100142	0.0					
DNLC Hits	702	1795228	100.0					
DNLC Longs	0	53	0.0					

You can also view the IO By File System and IO By Logical Volume reports from the Reports menu.

The Disk Queue graph shows the percentage of intervals during which there were 1-2, 3-4, 5-8, or more than 9 IO requests pending for the disk device over the cumulative collection time.

For more information, click on any highlighted topic:

[Network Information](#)

[Memory Information](#)

[CPU Information](#)

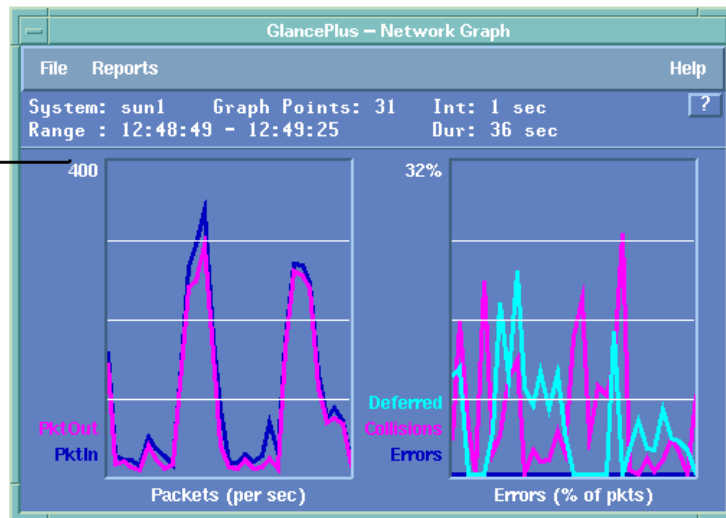
[System Information](#)

[Monitoring Applications](#)

Network Information

You can use several windows, available from the Reports menu in the GlancePlus Main window, to monitor network activity. You can also display the Network graph by clicking the Network button in the Main window.

The Network Graph is scaled based on the highest value ever encountered by GlancePlus.



In addition to this report, you can choose the NFS Global Activity, NFS By Operation and Network By Card graph reports from the Reports menu.

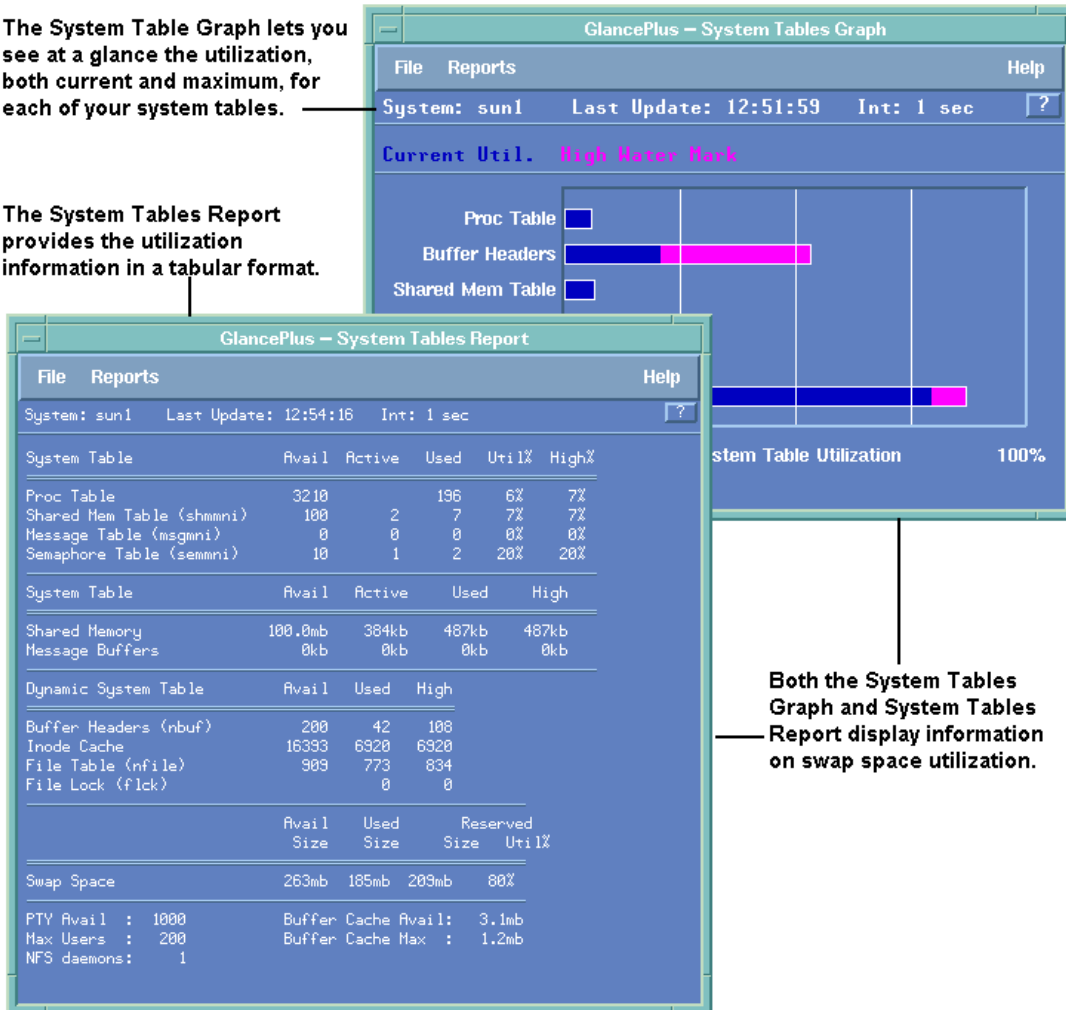
Interface Name	Network Type	In Pkt Rate	Out Pkt Rate	Coll Rate	Error Rate	Defer Rate	Input Pkt
lo0	Loop	26.3	26.3	0.0	0.0	0.0	29
le0	Lan	13.6	10.9	1.8	0.0	0.0	15

System Information

The following windows display information about your system tables. If you notice a table reaching maximum capacity, review its configuration.

The System Table Graph lets you see at a glance the utilization, both current and maximum, for each of your system tables.

The System Tables Report provides the utilization information in a tabular format.



Both the System Tables Graph and System Tables Report display information on swap space utilization.

Monitoring Applications

Applications are groups of processes that do similar work. For example, you might have an application called "payroll" for all the processes that perform payroll tasks. Use applications to easily monitor large groups of processes, and when needed, drill down to the process information for more specific details.

You define applications in a file external to GlancePlus, the "parm" file. The file and format are described in detail in Defining Applications. To display the Application List window, use the Reports menu in the Main window.

This window shows the applications defined by the application parm file and their corresponding metrics.

Double click an application from the list to see all the processes in that application.

GlancePlus – Application List

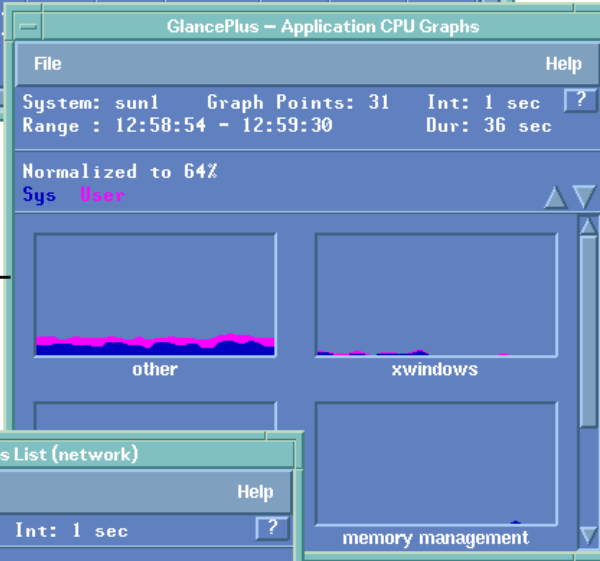
File Reports Configure Help

System: sun1 Last Update: 12:57:20 Int: 1 sec

Applications: All 5 Selected

App Name	Alive Proc	Active Proc	CPU %	I/O KB Rate	Block I/O Rt	Virtual Memory	Major Faults
other	43.0	2.0	9.6	15.4	0.0	73596	0
xwindows	11.0	0.0	0.0	0.3	0.0	30068	0
network	27.0	0.0	0.0	0.0	0.0	40192	0
memory_management	2						
other_user_root	109						

Use the Reports menu on the Application List window to view the Application CPU Graphs window. It displays a graph for each application's CPU.



GlancePlus – Application Process List (network)

File Reports Admin Configure Help

System: sun1 Last Update: 14:19:22 Int: 1 sec

Processes: All 26 Selected Application: network

Process Name	PID	CPU %	I/O Byte Rate	Block I/O Rate	Stop Reason	Pri
in.rlogind	51	0.0	0.0	0.0	SLEEP	52
rpcbind	103	0.0	0.0	0.0	SLEEP	58
inetd	120	0.0	0.0	0.0	SLEEP	48

Note that the order of the graphs reflects the order of the applications on the Application List. If the order changes on the Application List, the order of the graphs also changes.

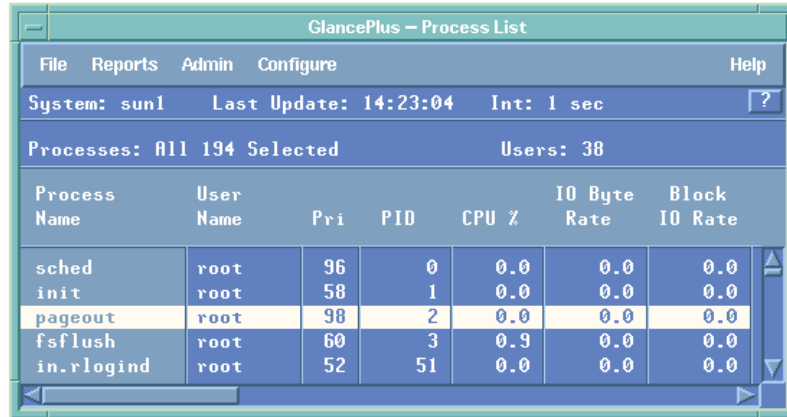
Managing Processes

A process represents a program executing on the system, which might contain hundreds or thousands of processes. GlancePlus provides you with the information you need to manage all the processes on your system. By using different GlancePlus windows, you can review an entire list of the processes. From this list, you can also filter and highlight specific interesting processes that require closer scrutiny.

To see a list of processes, select Process List from the Reports menu in the main window.

This window provides a complete list of the process names, user names, and associated metrics that define the load each process is putting on the system.

When you highlight a process from the list you have access to the options under the Reports and Admin menus.



The screenshot shows a window titled "GlancePlus - Process List". It has a menu bar with "File", "Reports", "Admin", "Configure", and "Help". Below the menu bar, it displays "System: sun1", "Last Update: 14:23:04", and "Int: 1 sec". It also shows "Processes: All 194 Selected" and "Users: 38". The main area is a table with the following columns: Process Name, User Name, Pri, PID, CPU %, IO Byte Rate, and Block IO Rate. The table lists several processes, with "pageout" highlighted in yellow.

Process Name	User Name	Pri	PID	CPU %	IO Byte Rate	Block IO Rate
sched	root	96	0	0.0	0.0	0.0
init	root	58	1	0.0	0.0	0.0
pageout	root	98	2	0.0	0.0	0.0
fsflush	root	60	3	0.9	0.0	0.0
in.rlogind	root	52	51	0.0	0.0	0.0

To rearrange report columns, sort fields, or filter certain processes to show on the Process list, use the Configure menu. See below for more information on these procedures.

Process Reports

The Reports menu provides more information about an individual process. Here are two of the available reports.

The Process Resources window shows CPU, disk, memory, and other specific measures concerning the process.

You can format the information by percentage or value, and by interval or since the process was created.

GlancePlus – Process Resources (gpm)					
File	Reports	Admin	Configure	Help	
System: sun1		Last Update: 14:26:08		Int: 1 sec	
Process Name	PID	PPID	User	State	
gpm	28235	28010	root	active	
Interval for collection: 1 sec					
Total CPU	:	7.0%	Priv. RSS (kb)	:	5380
User CPU	:	4.3%	Total RSS (kb)	:	6556
System CPU	:	2.6%	Data VSS (kb)	:	1176
Priority	:	59	Text VSS (kb)	:	7980
Nice Value	:	10	Stack VSS (kb)	:	28
Forced CSwitch	:	3	Total VSS (kb)	:	9184
Voluntary CSwitch	:	3	Minor Faults	:	0
Blocked On	:	SLEEP	Major Faults	:	0
Block Reads	:	0	Signals Received	:	0
Block Writes	:	0	System Calls	:	1085
Total IO Bytes	:	5kb	Threads Total	:	1

This window shows all the files a process has open. This may be useful in determining how many processes have a file open, and also the mode in which the file was opened.

GlancePlus – Process Open Files (gpm)				
File	Reports	Admin	Configure	Help
System: sun1		Last Update: 14:28:17		Int: 1 sec
Process Name	PID	PPID	User	State
gpm	28235	28010	root	active
Open Files: All 12 Selected				
File Name	File Nbr	File Type		
/dev/null	0	char		
/dev/pts/39	1	char		
/dev/pts/39	2	char		

More Process Reports

The following reports provide additional information about a process.

The Process Memory Regions window shows the detailed memory requirements of a process. This is useful when determining the specific types and sizes of memory regions that are required by the process.

Type	File Name	P/S	Ref Count
TEXT	<reg,ufs,inode:17121,/usr,/dev/dsk/c0t0d0s6>	Priv	6
BSS	<reg,ufs,inode:17121,/usr,/dev/dsk/c0t0d0s6>	Priv	6
DATA	<data>	Priv	1
LIBTXT	<reg,ufs,inode:142414,/usr,/dev/dsk/c0t0d0s6>	Priv	65

If, after reviewing the processes, you need to either kill a process or give it a new nice value, use the options on the Admin menu, available in the Process List window.

When you kill a process, the process is removed from the system. When you assign a process a new nice value, the rate at which its priority decays when being scheduled is altered.

Monitoring Your System with the Adviser

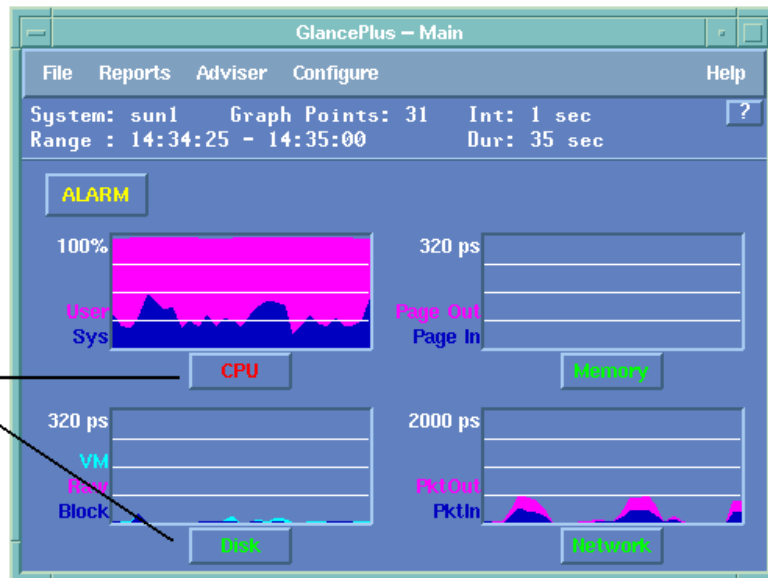
The GlancePlus Adviser automates monitoring and response to the state of key performance indicators. At each measurement interval, the Adviser executes a predefined set of instructions that examine performance metrics.

When a performance metric exceeds thresholds, the Adviser lights the ALARM button label or the CPU, Memory, Disk, or Network button labels.

This is the Alarm button. It is currently showing a warning with the yellow label.

These buttons notify you of events on your system through labels that change color.

The CPU button is showing a critical bottleneck.



Alarm Concepts

The ALARM statement is provided in the Adviser syntax to detect conditions that occur over a period of time, normally longer than one interval. By using the ALARM statement, you can specify actions that are executed at the onset of the alarm, actions that can be executed while the alarm condition remains true, and actions that can be performed when the alarm condition subsides.

The ALARM button is indicating a yellow alert.

The ALARM Statement defines the conditions that caused the label of the Alarm button to turn yellow.

When you press the ALARM button, the Alarm History window appears, explaining the condition that is causing the yellow alert.

The ALERT statement defines the message that will be displayed in the Alarm History window.

```

alarm CPU_Bottleneck > 50 for 2 minutes
start
  if CPU_Bottleneck > 90 then
    red alert "CPU Bottleneck probability= ", CPU_Bottleneck, "%"
  else
    yellow alert "CPU Bottleneck probability= ", CPU_Bottleneck, "%"
  repeat every 10 minutes
  if CPU_Bottleneck > 90 then
    red alert "CPU Bottleneck probability= ", CPU_Bottleneck, "%"
  else
    yellow alert "CPU Bottleneck probability= ", CPU_Bottleneck, "%"
  
```

Interval Time	Alarm Lvl	Message Text
14:43:11	HRN	CPU Bottleneck probability= 75.00%
14:43:18		End of CPU Bottleneck Alert
14:43:21	HRN	CPU Bottleneck probability= 75.00%
14:43:22		End of CPU Bottleneck Alert
14:43:23	HRN	CPU Bottleneck probability= 75.00%

Symptom Concepts

The SYMPTOM statement is provided in the Adviser to detect bottleneck situations that are influenced by more than one metric.

By summing the probabilities of the conditions of each rule found to be true, a single value is derived that represents the likelihood that a specific resource is OK, may be developing a problem, or is certainly experiencing a problem.

Each SYMPTOM statement in the Adviser generates a graph in the Symptom History Window to track the behavior of each symptom over a series of intervals. By observing the symptom graphs, you can determine if the bottleneck condition was isolated to one interval, or is a continuing trend that requires attention. You can also use the symptom values to update the color of the graph buttons on the Main window, to signal when a bottleneck condition is present.

In the picture below you can see what happens when a bottleneck is encountered on your system.

The CPU button is showing a critical alert because the probabilities in the SYMPTOM statement are above 90.

The Adviser syntax shows the Symptom statement, the statement's RULEs, and the probabilities.

The "type" defines the button on the main window that this symptom information will be tied to.

By looking at the Symptom History window, you can see how the Disk Bottleneck has been behaving over a period of intervals.

You can have as many graphs in the Symptom History window as you would like by defining them with the SYMPTOM statement.

A green alert indicates a probability of 0 to 50. Yellow indicates a probability of 51 to 90, and a red alert has a probability of 91 or greater.

Configuring GlancePlus

You can customize your GlancePlus display by changing various display characteristics, rearranging report columns, sorting report fields, and filtering processes.

Check Related Topics below for more information about configuring GlancePlus.

All configuration changes you make are saved for the next time you run GlancePlus, as long as you exit GlancePlus through the file menu or double click on the window menu bar. If you do not exit through the file menu, configuration changes are not saved. This includes sorting, filtering, column rearrangement, window size and location, and number of active windows.

If you never want to save configuration changes when you exit GlancePlus normally, use the **-nosave** command line option when you start GlancePlus.

Configure Colors

You can change the colors used for the graphs and process filters. If desired, you can also change the Adviser colors.

Depending on the availability of colors, GlancePlus may not be able to reserve its own private colors. If this is the case, and GlancePlus uses shared colors, you will not have access to the Configure Colors option.

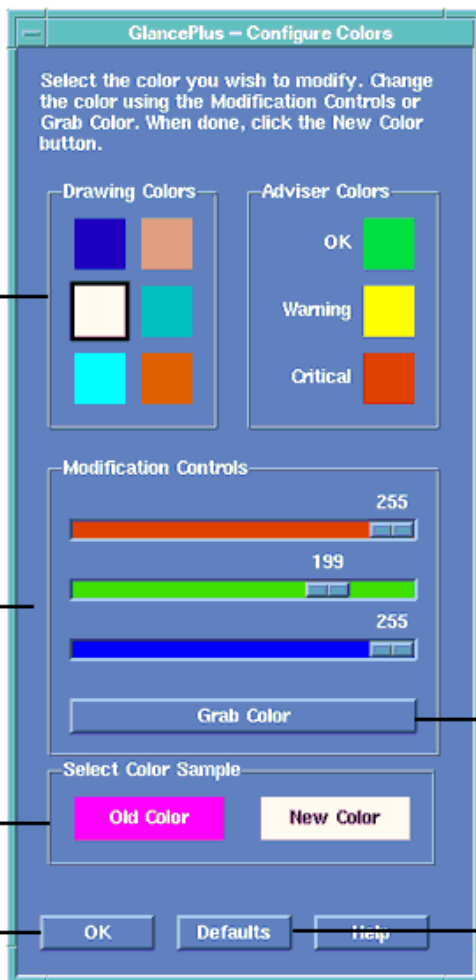
Choose Configure Colors from the Configure menu on the Main window.

Click the color in the Drawing Colors box or Adviser Colors box that you want to change.

Slide the controls in this area until the New Color button is the color you want.

Click New Color to make the change.

Click OK when you are done changing colors.



OR

Click the Grab Color bar, then click any color on your monitor screen. The change takes effect immediately.

Click the Defaults button to change all the colors back to their default values.

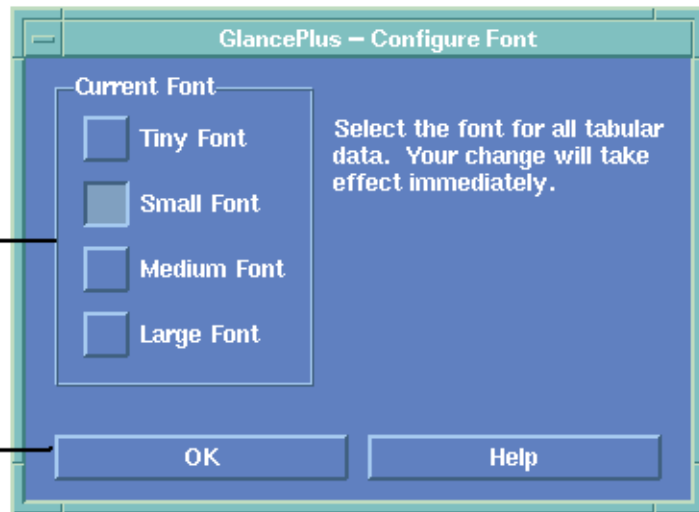
Configure Font

If you are viewing a report with many columns, decreasing the font size allows you to see more columns at one time. Changing the font size does not affect the font used to label graphs.

Choose Configure Font from the Configure menu on the Main window.

When you select a new font size, you immediately change the display font in the window headers and the various report windows.

Click OK.



Reducing the font size allows you to see more columns of a report at one time.

Configure Measurement

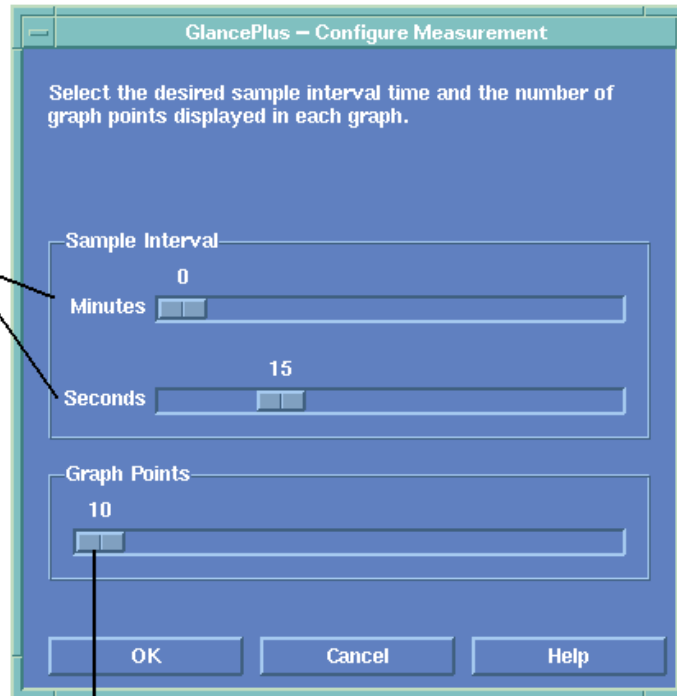
With GlancePlus you can define the length of the measurement interval and the number of these intervals to save into the history buffers. The interval defines the time between each measurement update. The history buffers save the pertinent interval information for use in graphs and the Application History window. For graphs that show history, the history buffer dictates the number of points shown on the graph, with each point representing an interval.

Choose **Configure Measurement** from the **Configure** menu on the main window.

Slide the control to the left or right to increase or decrease the interval time.

When you increase interval time, there is less overhead on the system because the measurements are further apart. This could hide certain data spikes, however, as the activity is averaged over the interval.

Decreasing the interval puts more overhead on the system, but provides a more accurate report of the activity over the interval.



Similarly, slide the control to change the number of graph points shown on a graph.

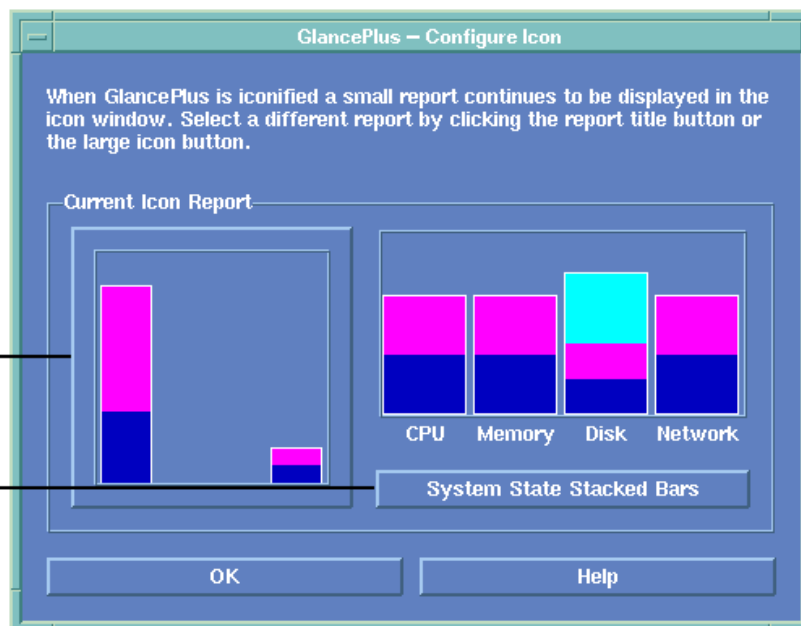
When you increase the number of graph points per graph you see a larger timespan. Increasing the graph points slightly increases memory demands on GlancePlus.

Configure Icon

Choose **Configure Icon** from the **Configure** menu on the Main window.

To change the type of graph displayed when you iconize GlancePlus, click either here...

or here.



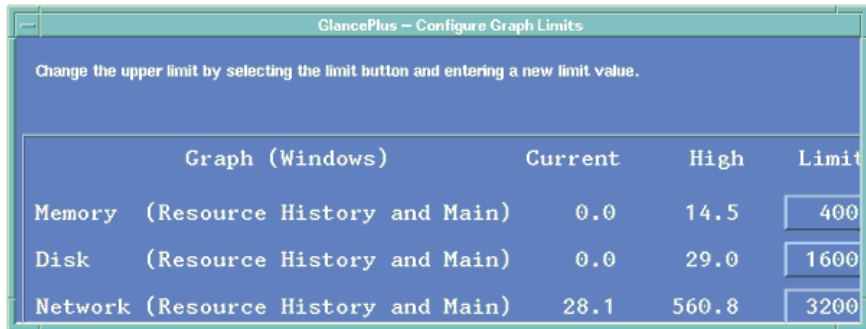
Configure Graph Limits

The Disk and Network graphs automatically scale up if a data point is measured that is higher than the current high value (which is the highest value that GlancePlus has observed since it was installed). If the top point of these graphs is an unusual spike, you may want to reset the limit to a lower value.

Choose Graph Limits from the Configure menu in the Main window.

Change the Y - axis scale by clicking the appropriate Limit button and typing a new limit.

Click OK when you are done. The graph scale automatically reflects the new limit.



The limit value you enter must always be equal to or higher than the highest measured point found in the history buffer. If a point in the history buffer is higher than the input value, the highest measured point is used as the limit value.

Configure Process Filters

Choose Filters from the Configure menu on the Process List window.

Locate the first process metric you want to filter. Use the scrollbar at the bottom of the window to find the metric.

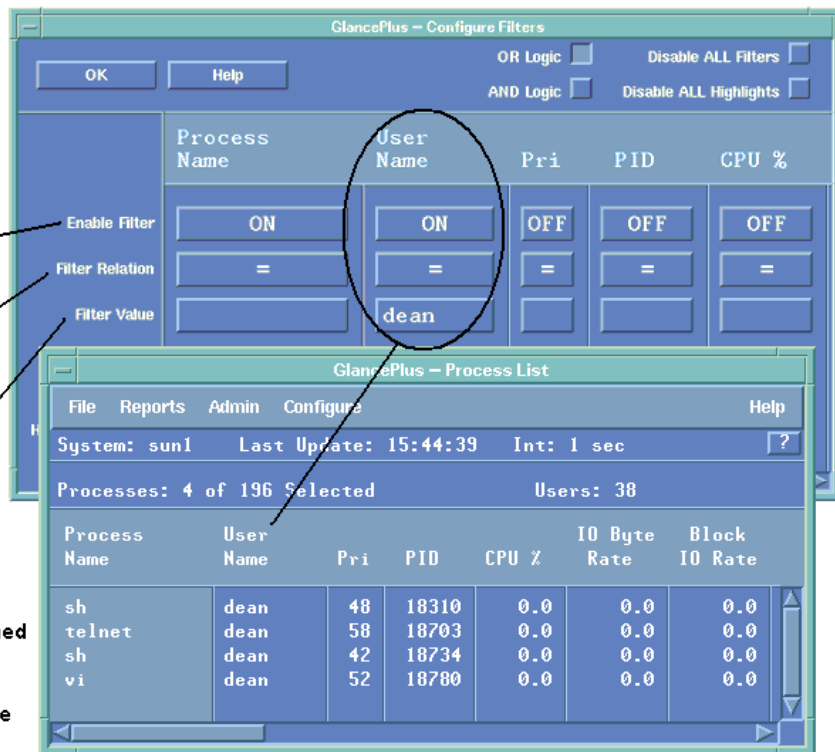
Click the Enable Filter box for the metric you are filtering.

Click the Filter Relation box until the relational symbol you want is displayed.

Click the Filter Value box, then type a value. If you don't know the type of value you can enter for a specific metric, check the help for the Process List window.

Click OK when you are finished setting filter conditions.

The Process List now shows only those processes with the User Name = dean.

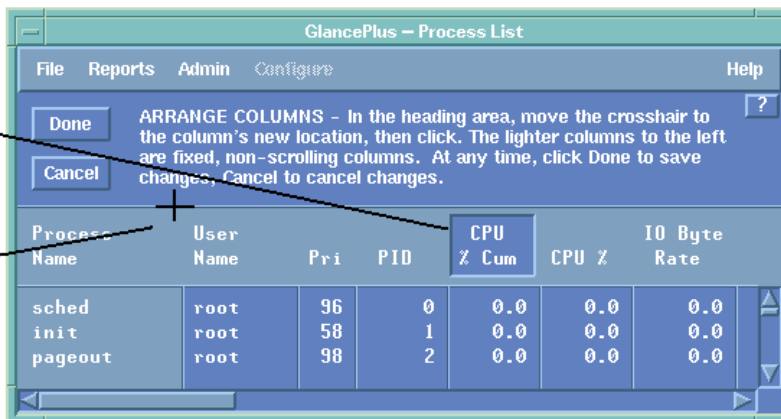


Rearrange Report Columns

To move the CPU % Cum column, we clicked the Done button...

then moved the crosshair to the lighter shaded, non-scrolling region to the left.

When we click next, the CPU % Cum column will appear in its new location.



Sort Columns

Choose Sort Fields from the Configure menu.

To sort the processes by CPU %, select it and move the crosshair cursor to the "No Sort Fields" column, then click.

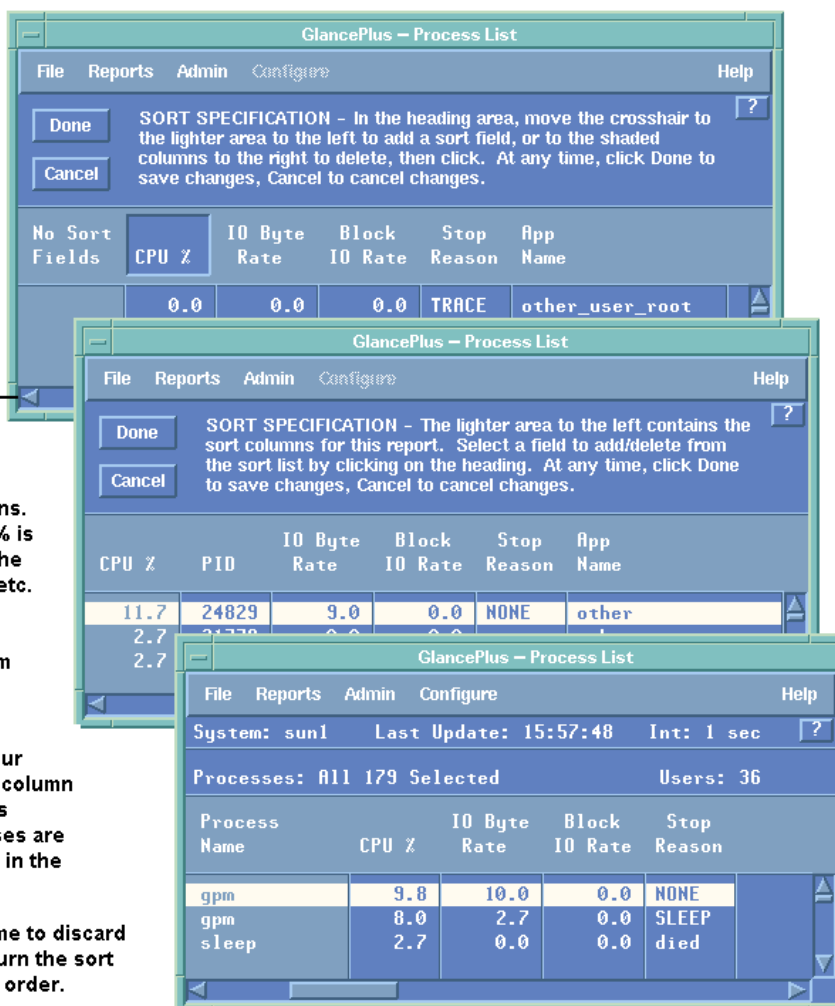
If a column isn't visible, move the cursor to the scrollbar and scroll until the desired column appears.

CPU % now appears under the sort columns. Notice that the CPU % is now sorted to show the highest CPU % first, etc.

Continue adding or deleting columns from the sort area.

Click Done to save your changes. The CPU % column returns to its previous position. The processes are now sorted by CPU % in the Process List Window.

Click Cancel at any time to discard your changes and return the sort fields to their original order.



Guided Tour

The GlancePlus Guided Tour is a quick introduction to the features of GlancePlus.

Because the illustrations in this tour are large, you may want to enlarge your help window before you begin.

Click one of the topics below to begin the tour.

[Guided Tour - GlancePlus Main Window](#)

[Guided Tour - CPU Bottleneck, panel 1](#)

[Guided Tour - Memory Bottleneck, panel 1](#)

[Guided Tour - Configuring GlancePlus](#)

[Guided Tour - Alarms & Symptoms](#)

Guided Tour - GlancePlus Main Window

The Main window is the first window you see when you run GlancePlus. Its menus provide access to all GlancePlus features. If you wish, you can navigate using speed keys as an alternative to using the mouse.

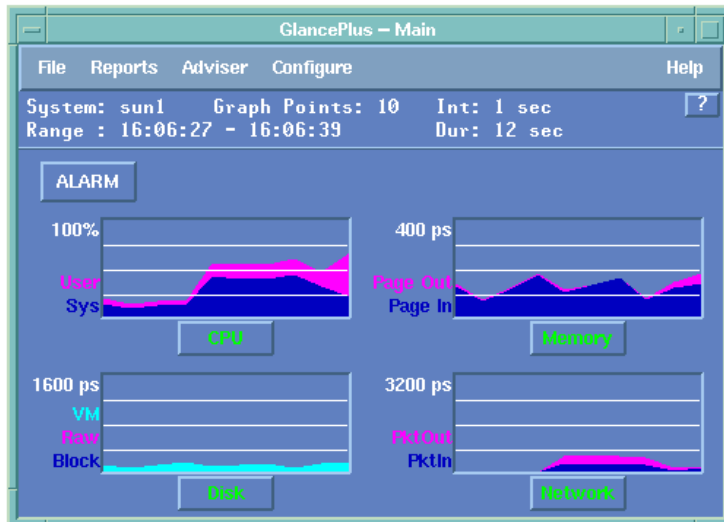
These four menus provide access to all GlancePlus features.

The ALARM button label changes color when selected performance metrics exceed thresholds you define.

These button labels also change color (from green to yellow or red) based on the symptoms you defined in the Adviser.

Click the button to see a secondary graph showing activity for that area.

These graphs show system level activity.



Guided Tour - CPU Bottleneck, panel 1

The following panels walk you through a typical CPU bottleneck scenario, from the time you notice there's a potential problem to the discovery of the cause of the bottleneck.

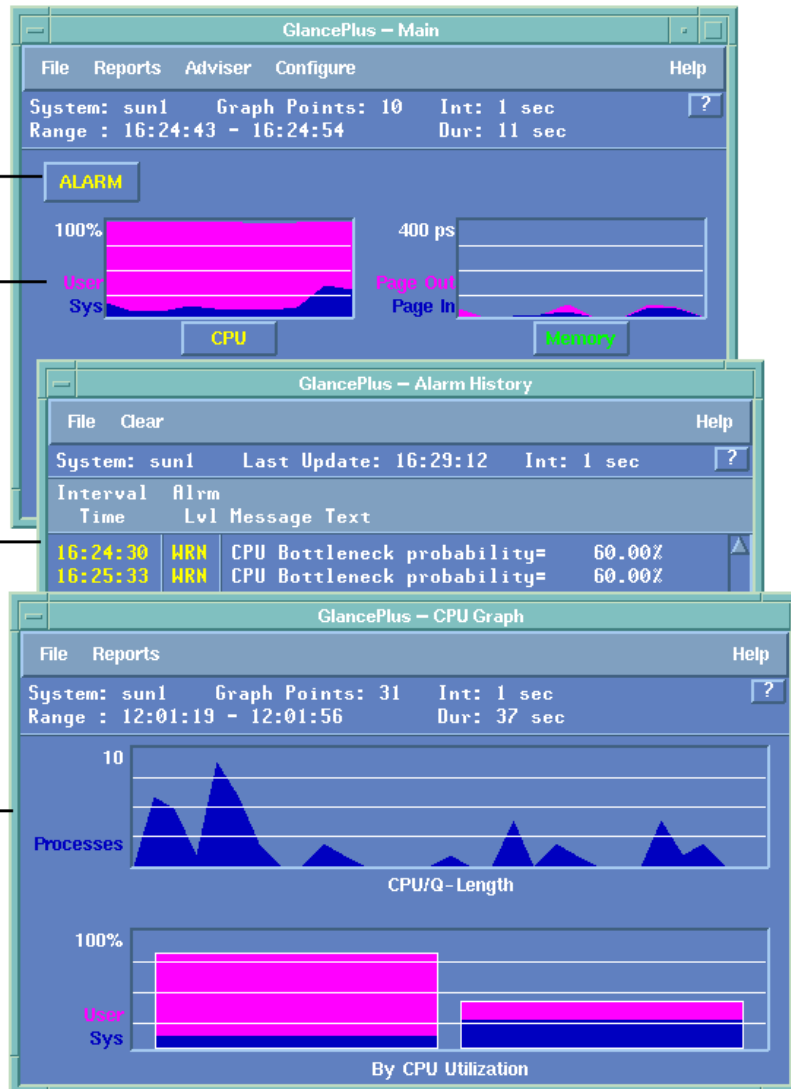
The ALARM button is yellow, indicating a warning condition has occurred.

The CPU graph indicates that almost all of the CPU resources are being monopolized by User.

The Alarm History window displays when you click the ALARM button. It shows the history of alarm activity on the system.

In this example you see that there is a 75% probability of a CPU bottleneck.

The CPU Graph displays when you click the CPU button on the Main window. This graph shows an increase in the load average over time which means more processes will be contending for the CPU.



Guided Tour - CPU Bottleneck, panel 2

Applications are processes you group together to better organize your picture of system activity. In this example we'll look for the application and the process using the majority of the CPU, causing a bottleneck.

This window displays when you select Application List from the Reports menu on the Main window.

We sorted by CPU % to show the application using the highest percentage of CPU.

The application 'other user root' is using 13.4 % of current CPU.

Double click the application name to see which processes are in that application.

In this example we see that the process 'cpuhog' is using 96.5 % of the current CPU.

GlancePlus - Application List

File Reports Configure Help

System: sun1 Last Update: 16:39:16 Int: 1 sec ?

Applications: All 5 Selected

App Name	Alive Proc	Active Proc	CPU %	IO KB Rate	Block IO Rt
other_user_root	102.8	6.8	13.4	300.9	0.0
other	40.0	2.0	34.8	63.6	0.0
network	24.0	0.0	0.0	0.0	0.0
xwindows	10.0	1.0	0.9	0.9	0.0

GlancePlus - Application Process List (other_user_root)

File Reports Admin Configure Help

System: sun1 Last Update: 13:25:20 Int: 1 sec ?

Processes: All 110 Selected Application: other_user_

Process Name	CPU %	IO Byte Rate	Block IO Rate	Stop Reason	Pri	User Name
ksh	0.0	0.0	0.0	SLEEP	46	root
sh	0.0	0.0	0.0	SLEEP	48	root
cpuhog	96.5	0.0	0.0	PRI	0	root
sh	0.0	0.0	0.0	SLEEP	40	root
sh	0.0	0.0	0.0	SLEEP	29	root
ksh	0.0	0.0	0.0	SLEEP	48	root

Having identified the offending process, as system administrator, you can take the appropriate action.

Guided Tour - Memory Bottleneck, panel 1

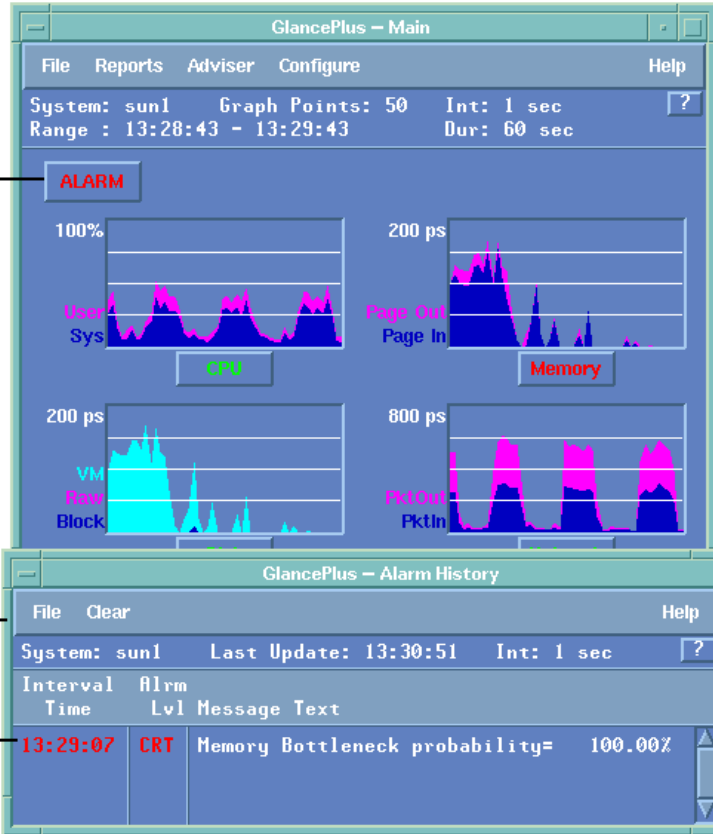
This is the first of two panels that walk you through a typical memory bottleneck scenario, from the time you notice there's a potential problem to the discovery of the cause of the bottleneck.

The ALARM button label is red, indicating a critical condition has occurred.

The main Memory graph shows that memory pressure is causing a high page out rate.

This window displays when you click the ALARM button. It shows alarm activity.

In this example you see that the memory bottleneck probability is at 100 %.



Let's look at the process level for further clues . . .

Guided Tour - Memory Bottleneck, panel 2

Next we'll look at the process level to find the processes using more than their share of memory. The process level is where your search for bottlenecks will frequently end.

This window displays when you choose Process List from the Reports menu on the Main window.

We sorted by Res Mem (resident memory) and Virtual Memory.

These two memhog processes are using a very high number of kilobytes of both resident memory and virtual memory.

GlancePlus - Process List				
File Reports Admin Configure Help				
System: sun1		Last Update: 13:45:45		Int: 1 ?
Processes: All 178 Selected			Users: 32	
Process Name	TTY	Res Mem	Virtual Memory	
memhog	pts/20	16988	16988	
memhog	pts/20	16988	16988	
in.telnetd		404	1388	

To highlight those processes using the most memory, we selected Highlight Values and Highlight Colors for Res Mem and Virtual Memory.

We have found the processes that are the root of the memory problem.

GlancePlus - Configure Filters					
OK		Help		OR Logic <input type="checkbox"/>	Disable ALL Filters <input type="checkbox"/>
		AND Logic <input type="checkbox"/>		Disable ALL Highlights <input type="checkbox"/>	
	PPID	TTY	Res Mem	Virtual Memory	Priv RSS
Enable Filter	OFF	OFF	OFF	OFF	OFF
Filter Relation	=	=	=	=	=
Filter Value					
Enable Highlight	OFF	OFF	ON	ON	OFF
Highlight Relation	=	=	>	>	=
Highlight Value			5000	12000	
Highlight Color	Red	Red	Magenta	Blue	Red

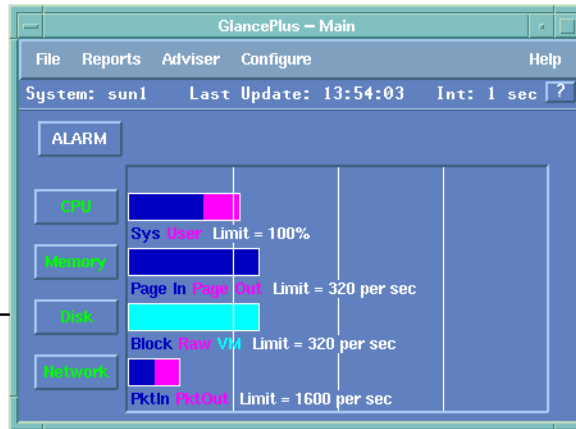
Having identified the offending processes, as system administrator, you can take the appropriate action.

Guided Tour - Configuring GlancePlus

Use the Configure menu on the Main window to set graph colors, font size, measurement interval, GlancePlus icon, graph limits, and the main graph style. The windows below show some examples of the GlancePlus configuration options.

You can change the style of the main graphs. Choose from horizontal bars (shown here), vertical bars, pie charts, or resource history graphs.

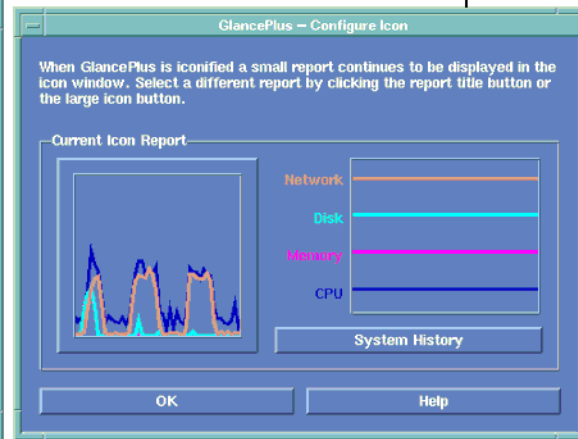
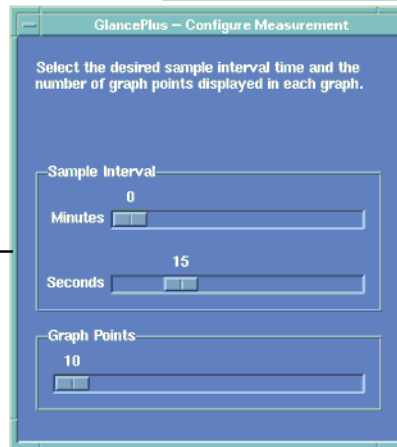
Use Configure Graph Limits to set new upper limits for the Disk and Network graphs.



This Main window example shows the default colors and the small font. You can change the graph colors and the font size, if desired.

Use the Configure Icon window to change the icon that is displayed when you iconify GlancePlus.

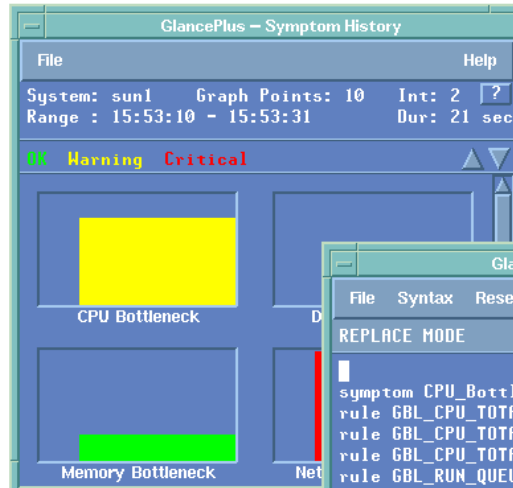
You can also change the measurement interval and number of graph points.



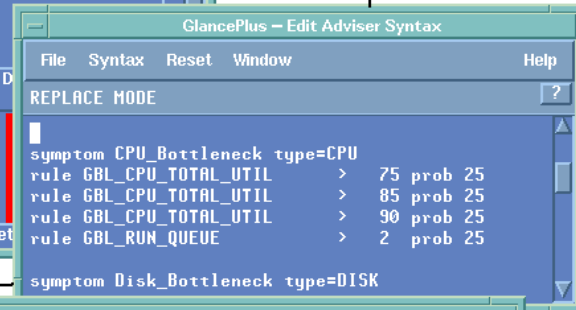
Guided Tour - Alarms & Symptoms

The GlancePlus Adviser monitors your system. It works in two ways: by lighting an alarm button when conditions occur that you specify, and notifying you of symptoms for potential CPU, disk, network, or memory bottlenecks.

This window shows bottleneck history for the bottleneck symptoms you defined in the Symptom Syntax window.



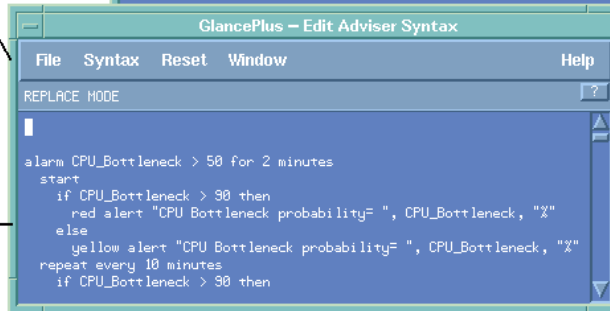
GlancePlus provides pre-defined bottleneck symptoms. The window below shows the default symptom syntax for a CPU bottleneck.



Open the Adviser Syntax window from the Adviser menu on the Main window. You can toggle between the Symptom window and the Alarm window.

Use the Alarm window to set your alarms. This window shows an example of the syntax for a CPU alarm.

Note that the `cpu_bottleneck` symptom from the Symptom window is used to create an alarm definition.



Customize GlancePlus Start-Up

You can customize your operation of GlancePlus by adding command line options to the `xglance` command. You can add as many parameters to the `xglance` command line as you like. Here are the three most commonly-used command line options:

`-nosave` By default, when GlancePlus closes, it automatically saves all configuration changes you made during that session, including sorting, filtering, column rearrangement, window size and location, and number of active windows. If you don't want to save these options, start GlancePlus with the `-nosave` option.

```
xglance -nosave ...
```

`-sharedclr` If you will be running several versions of GlancePlus or just want to share colors with another application, use the `-sharedclr` option. It causes GlancePlus to use a color scheme that is shared by other X-window applications that are currently running. While it disables the ability to configure colors within GlancePlus, it potentially solves the problem of applications being unable to run because of a shortage of colors.

```
xglance -sharedclr ...
```

`-rpt` Use the `-rpt` option to tell GlancePlus to open specific windows on startup (this is in addition to the windows that were saved automatically the last time you ran GlancePlus). For example:

```
xglance -rpt windowname1 windowname2 ...
```

where `windowname1` and `windowname2` are the GlancePlus window names for the windows you want to display. Refer to the Windows List for window names for all the GlancePlus windows you can display at startup.

Override Configuration Updates

You can set up GlancePlus so that certain windows display when you open GlancePlus.

1. Arrange the windows on your screen in the order that you want to see them.
2. Exit GlancePlus. This saves the configuration you set up as your own *control panel*.
3. The configuration changes are saved in the `.gpmhp <hostname>file`, where `hostname` is the name of the system (`uname -n`) where `xglance` is running.
4. When you restart GlancePlus, use the `-nosave` option. This ensures that what was saved in your configuration file will not be changed by anything you do in future sessions of GlancePlus and your *control panel* arrangement will not change.

Windows List

Use the GlancePlus window name when you want to start GlancePlus with specific windows already open. Use the following command:

```
xglance -rpt windowname1 windowname2 ...
```

where `windowname1` and `windowname2` are GlancePlus window names from the right-hand column of the list below.

NOTE:

Some windows in the list are specific to particular platforms, as indicated in parentheses.

<u>Window</u>	<u>GlancePlus Window Name</u>
GlancePlus Main	Main
Alarm History	AlarmHistory
Application CPU Graphs	ApplicationCpuGraphs
Application List	ApplicationList
CPU By Processor	CpuByProcessor
CPU Graph	CpuGraph
CPU Report	CpuReport
Disk Graph	DiskGraph
Disk Queue Graphs (Sun, SNI)	DiskQueueGraphs
Disk Report	DiskReport
File System Capacity (Sun, SNI)	FileSystemCapacity
IO By Disk	IOByDisk
IO By File System (N/A on SNI and Sun)	IOByFileSystem
IO By Logical Volume (HP-UX 11, Sun)	IOByLogicalVolume
IO By Virtual Disk (SNI only)	IOByVirtualDisk
Memory Graph	MemoryGraph
Memory Report	MemoryReport
Memory Usage Graph	MemoryUsageGraph
Network By Card Graph	NetworkByCardGraph
Network By Interface	NetworkByInterface
Network By LAN	NetworkByLan
Network Graph	NetworkGraph
NFS By Operation	NfsByOperation
NFS By System (HP-UX)	NfsBySystem
NFS Global Activity	NfsGlobalActivity
PRM CPU Graphs (HP-UX 10.20, 11.x)	PRMCPUGraphs
PRM Group List (HP-UX 10.20, 11.x)	PRMGroupList
PRM Memory Graphs (HP-UX 10.20, 11.x)	PRMMemgraphs
Process List	ProcessList

Process Memory Regions	ProcessMemoryRegions
Resource History	ResourceHistory
Swap Space	SwapSpace
Symptom History	SymptomHistory
Symptom Status	SymptomStatus
System Attributes	SystemAttributes
System Calls (HP-UX 10.20, 11.x)	SystemCalls
System Tables Graph	SystemTablesGraph
System Tables Report	SystemTablesReport
Transaction Tracking	TransactionTracking
Virtual Memory Graph (AIX, NCR)	VirtualMemoryGraph
Wait Queue Graphs (Sun, AIX)	WaitQueueGraphs
Wait States (N/A on Sun)	WaitStates

Set GlancePlus Measurements

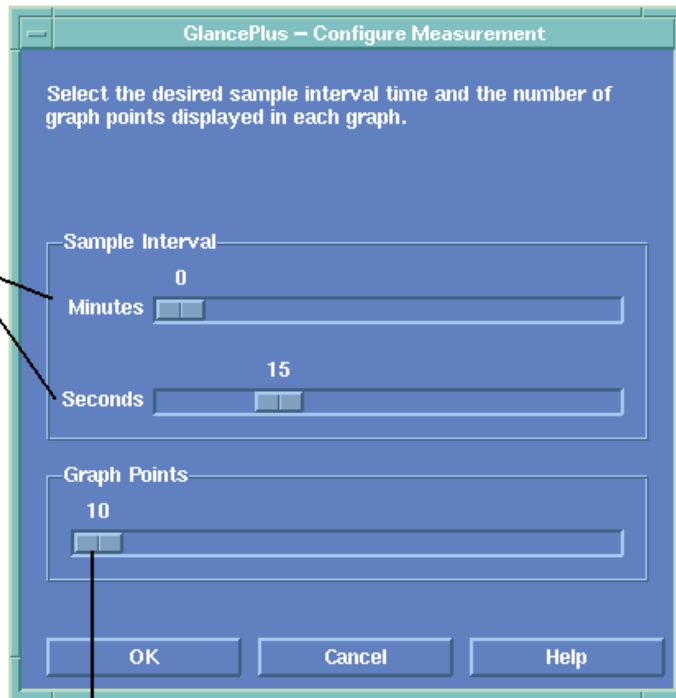
You can set the collection interval and the number of graph points displayed in a graph. For example, you might want to monitor activity over a short period of time by setting a small interval. If you want to monitor a longer period of time, you would set a longer collection interval.

Choose Configure Measurement from the Configure menu on the main window.

Slide the control to the left or right to increase or decrease the interval time.

When you increase interval time, there is less overhead on the system because the measurements are further apart. This could hide certain data spikes, however, as the activity is averaged over the interval.

Decreasing the interval puts more overhead on the system, but provides a more accurate report of the activity over the interval.



Similarly, slide the control to change the number of graph points shown on a graph.

When you increase the number of graph points per graph you see a larger timespan. Increasing the graph points slightly increases memory demands on GlancePlus.

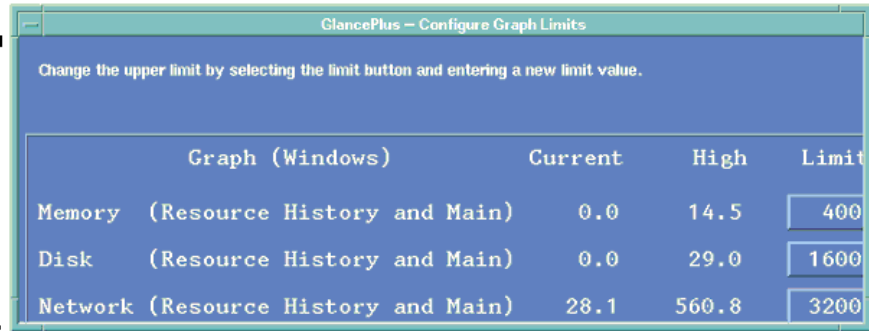
Change Graph Limits

To adjust the scaling for the Disk and Network graphs on the Main and Resource History windows:

Choose Graph Limits from the Configure menu in the Main window.

Change the Y - axis scale by clicking the appropriate Limit button and typing a new limit.

Click OK when you are done. The graph scale automatically reflects the new limit.



The new limit must be equal to or higher than the highest amount in the history buffer. If you set a lower limit, GlancePlus automatically readjusts the limit to the High value to avoid clipping any data.

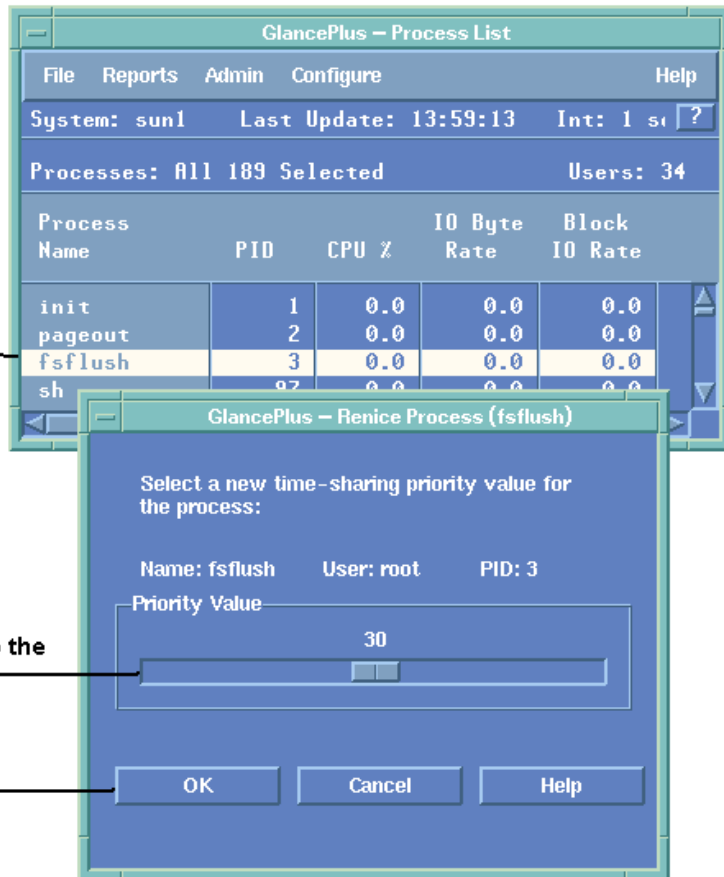
Renice a Process

Click the process name you want to change.

Choose Renice Process from the Admin menu.

Slide the Priority Value scale to the left or right to select a new time-sharing priority value.

Click OK.



You can also renice a process from the Admin menu of any Process detail window.

Decreasing the nice value can increase a process' chance of acquiring CPU time. You make that process "not as nice," since it leaves less CPU time for other processes. Increasing the nice value lessens the process' chance of getting CPU time.

Adjusting the nice value has the biggest impact on processes that are CPU-bound. On non CPU-bound processes, it may have little or no effect.

Reset Cumulative Values

To reset cumulative values to zero, choose one of the following options:

Select Reset Cum to Zero from the File menu.

OR

Type <Ctrl>z.

In this example, the Cum Time values will be reset to zero.

GlancePlus - CPU Report

System: sun1 Last Update: 14:03:34 Int: 1 sec

Top CPU User: lockd PID: 125 CPU Util: 62.7%

State	Current	Avg.	High	Time	Cum Time
User	11.8%	13.9%	71.2%	0.1	397.5
System	37.3%	19.0%	64.3%	0.4	544.1
Idle	50.9%	67.2%	96.4%	0.6	1929.0

Activity Rate C Rate High

Syscalls	10119.0	5442.9	12637.2
Interrupts	754.5	693.3	12766.3
Cont Switches	383.6	295.7	1416.3

Queues Length Avg. High

CPU Queue	0.0	0.1	6.0
Load Average	0.6	0.8	1.7

Terminate a Process

Select the process you want to terminate.

Choose Kill Process from the Admin menu.

Click OK. You must be the owner to terminate a process.

GlancePlus - Process List

System: sun1 Last Update: 14:07:36 Int: ?

Processes: All 178 Selected Users:

Process Name	User Name	Pri	PID	CPU %
pageout	root	98	2	0.0
fsflush	root	60	3	0.0

GlancePlus - Kill Process (pageout)

Kill Process

Name: pageout User: root PID: 2

OK Cancel Help

You can also terminate a process from any process detail window.

GlancePlus uses the SIGKILL (kill -15) signal to kill a process.

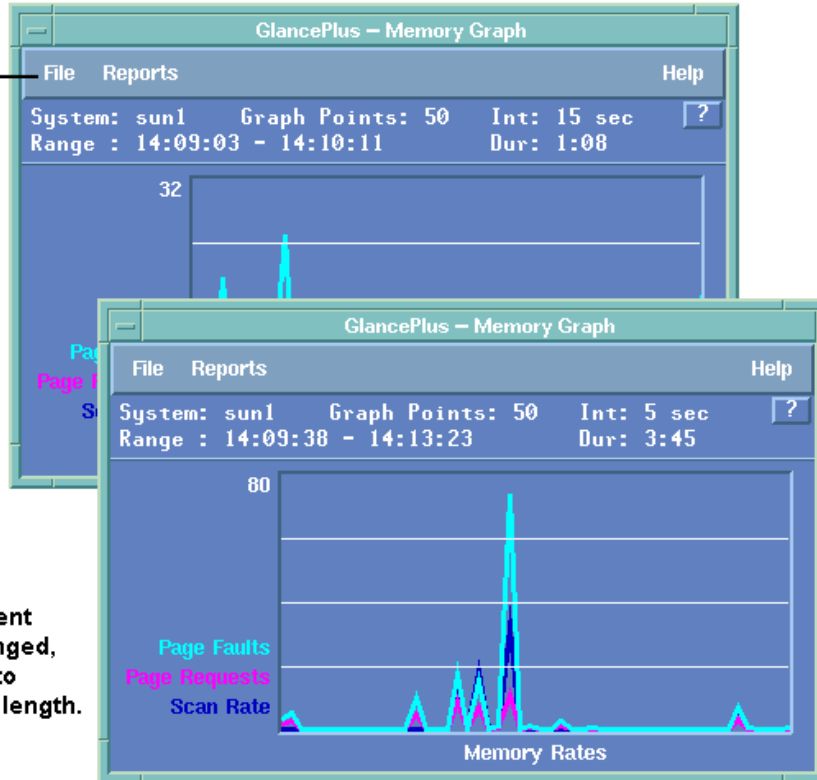
Update Metrics

You can update the metrics on the screen at any time. For example, even though the defined interval is 15 seconds, you can request an update before 15 seconds has elapsed.

Choose Update Now from the File menu.

OR

Type <Ctrl>u.



The defined measurement interval remains unchanged, but the graph changes to reflect the new interval length.

Change Display Colors

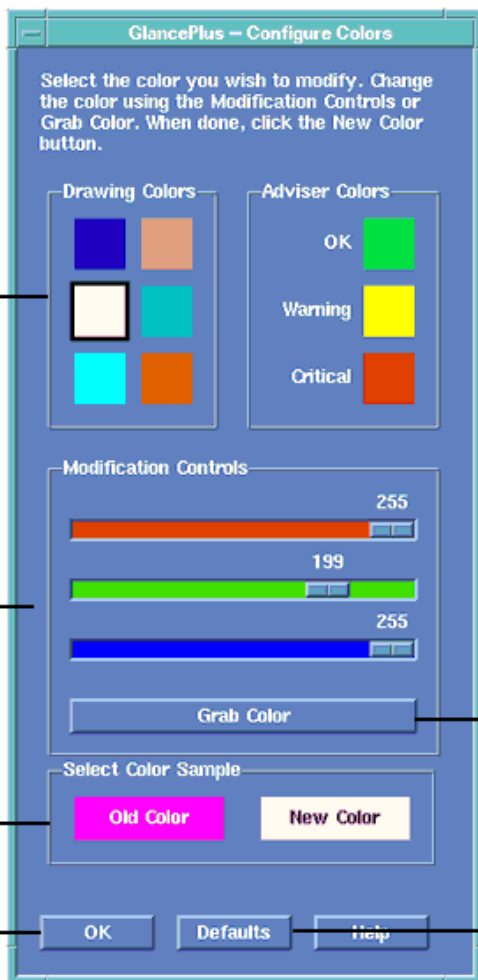
Choose Configure Colors from the Configure menu on the Main window.

Click the color in the Drawing Colors box or Adviser Colors box that you want to change.

Slide the controls in this area until the New Color button is the color you want.

Click New Color to make the change.

Click OK when you are done changing colors.



OR

Click the Grab Color bar, then click any color on your monitor screen. The change takes effect immediately.

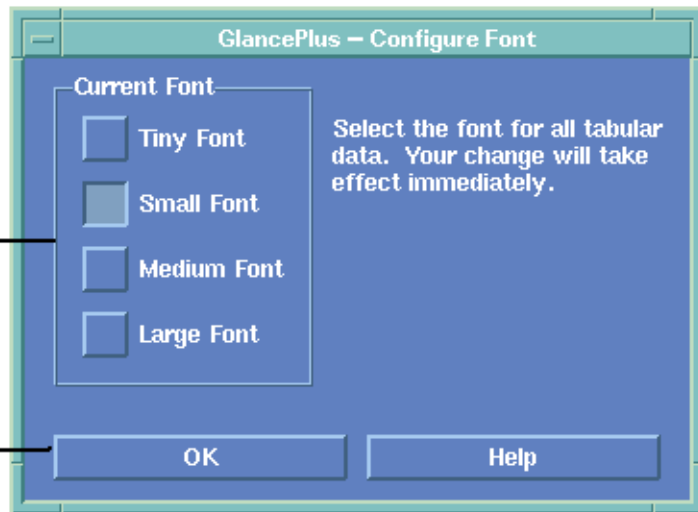
Click the Defaults button to change all the colors back to their default values.

Change Font Size

Choose Configure Font from the Configure menu on the Main window.

When you select a new font size, you immediately change the display font in the window headers and the various report windows.

Click OK.



Reducing the font size allows you to see more columns of a report at one time.

Change Graph Icon

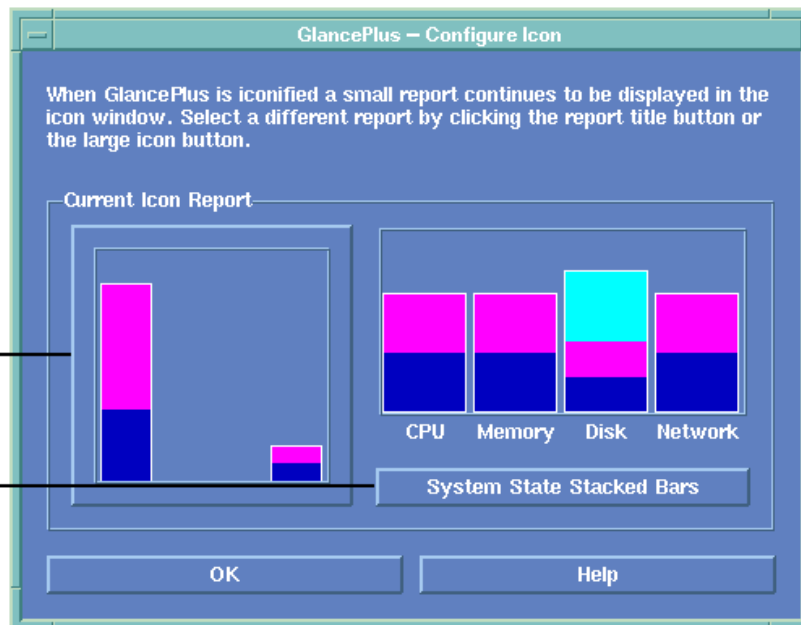
You can change the way the graph displays when GlancePlus is iconified. When iconified, GlancePlus continues to report your system's activity through the icon. The icon border also changes color when either a red or a yellow alert occurs.

If your GlancePlus icon displays too large or too small, you can override the default icon size by changing the variables in the `Gpm` file. For more information on editing the `Gpm` file, see [Modifying Runtime Parameters](#).

Choose Configure Icon from the Configure menu on the Main window.

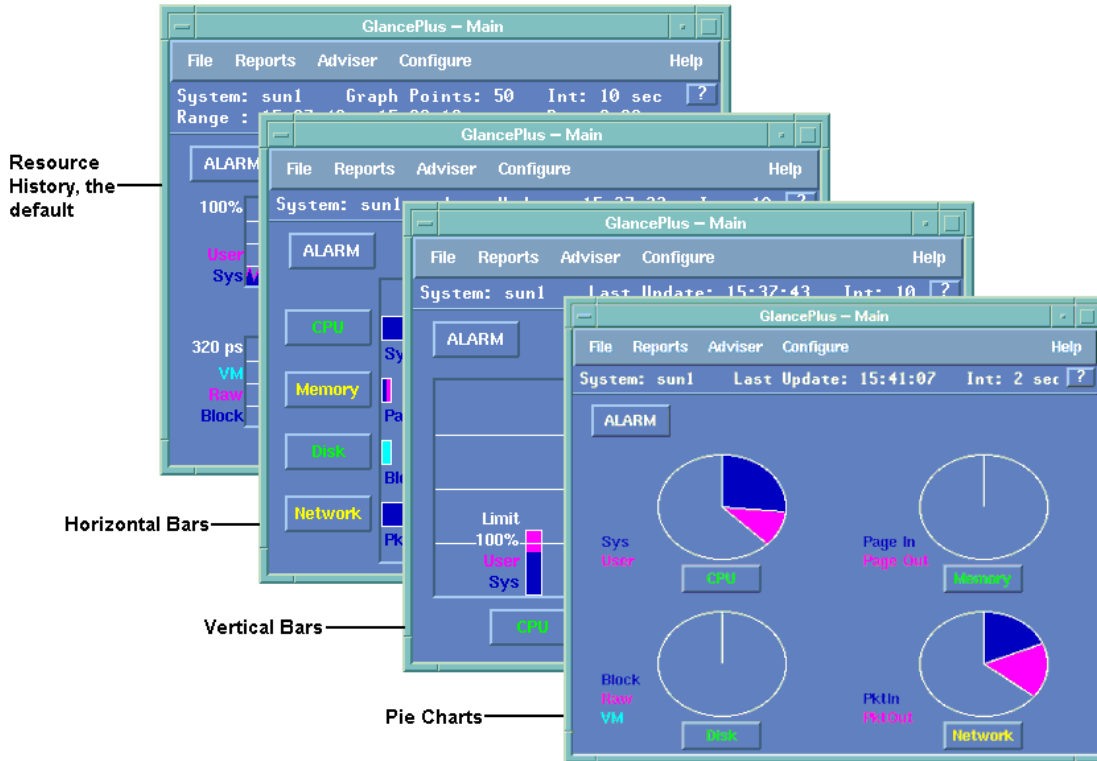
To change the type of graph displayed when you iconize GlancePlus, click either here...

or here.



Change Main Graph Style

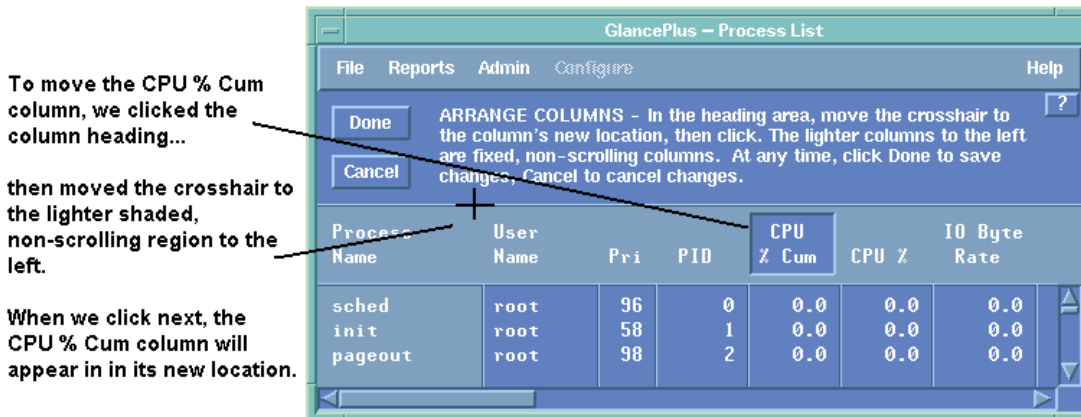
You can display the information in your main window in one of four ways:



Select Main Graph from the Configure menu and choose one of these styles.

Rearrange Report Columns

To rearrange report columns:



Click Cancel at any time to cancel your changes and return the columns to their original order.

Sort Report Fields

You can change the way information is sorted in a report window by adding or deleting columns from the sort area. After you have selected sort columns, sorting is done from left to right as specified by the columns in the sort area, with the left-most column sorted first.

To change the sort order:

Choose Sort Fields from the Configure menu.

To sort the processes by CPU %, select it and move the crosshair cursor to the "No Sort Fields" column, then click.

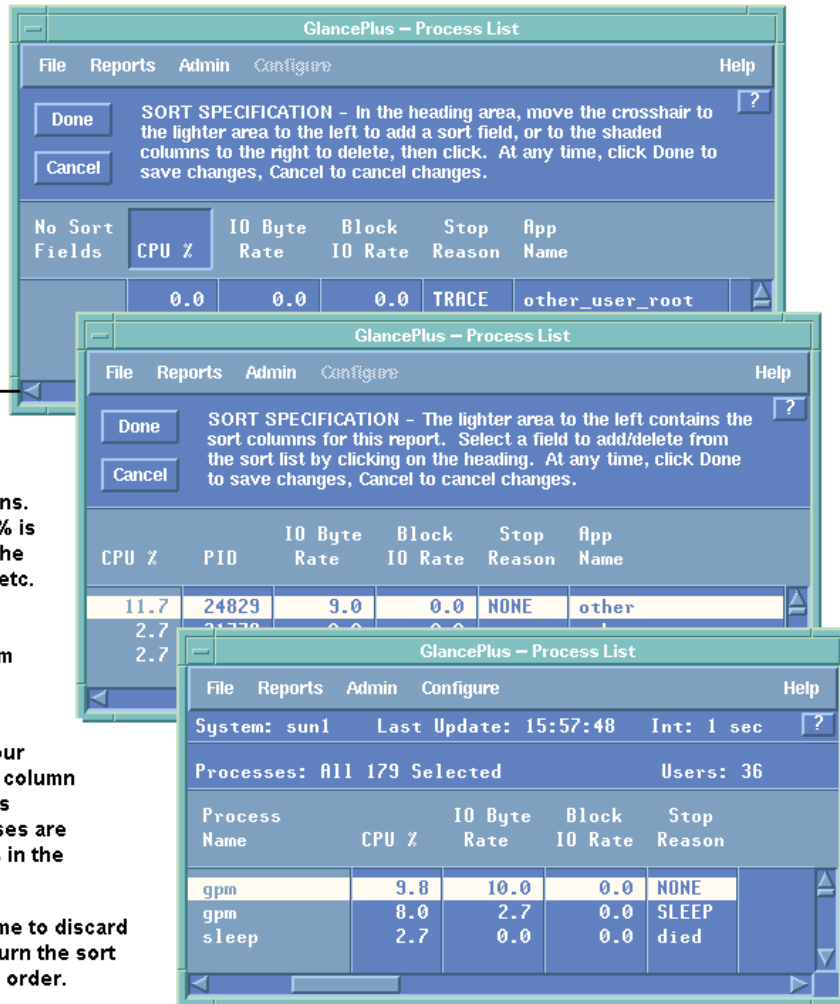
If a column isn't visible, move the cursor to the scrollbar and scroll until the desired column appears.

CPU % now appears under the sort columns. Notice that the CPU % is now sorted to show the highest CPU % first, etc.

Continue adding or deleting columns from the sort area.

Click Done to save your changes. The CPU % column returns to its previous position. The processes are now sorted by CPU % in the Process List Window.

Click Cancel at any time to discard your changes and return the sort fields to their original order.



If you'd like to arrange the columns differently, see Rearrange Report Columns.

Set Filters

To set filters so that a process displays in the Process List window only when certain conditions are met:

Choose Filters from the Configure menu on the Process List window.

Locate the first process metric you want to filter. Use the scrollbar at the bottom of the window to find the metric.

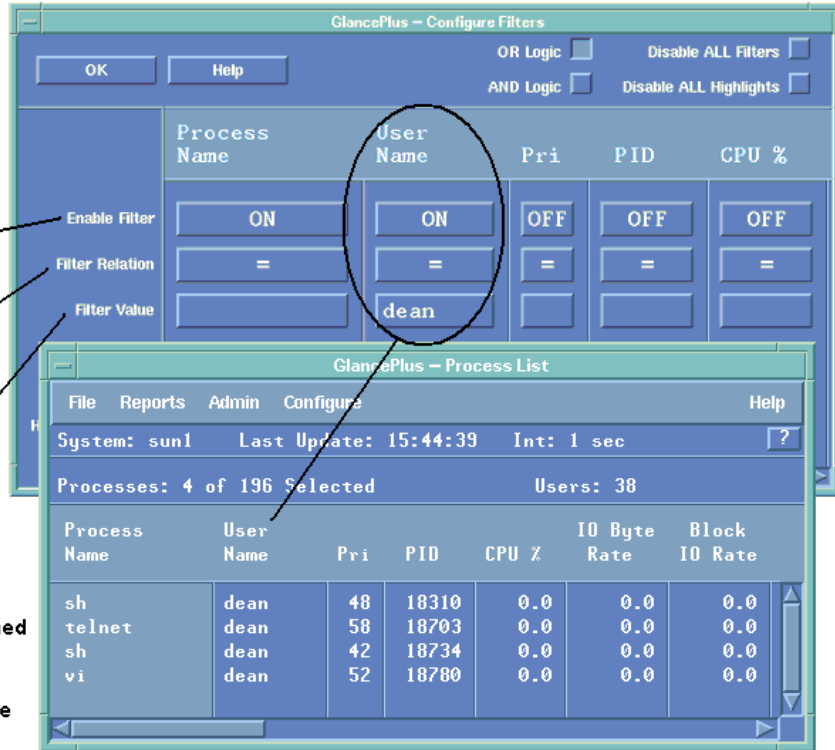
Click the Enable Filter box for the metric you are filtering.

Click the Filter Relation box until the relational symbol you want is displayed.

Click the Filter Value box, then type a value. If you don't know the type of value you can enter for a specific metric, check the help for the Process List window.

Click OK when you are finished setting filter conditions.

The Process List now shows only those processes with the User Name = dean.



Click OR Logic to display processes in the Process List window that match any one or more of the criteria you specify.

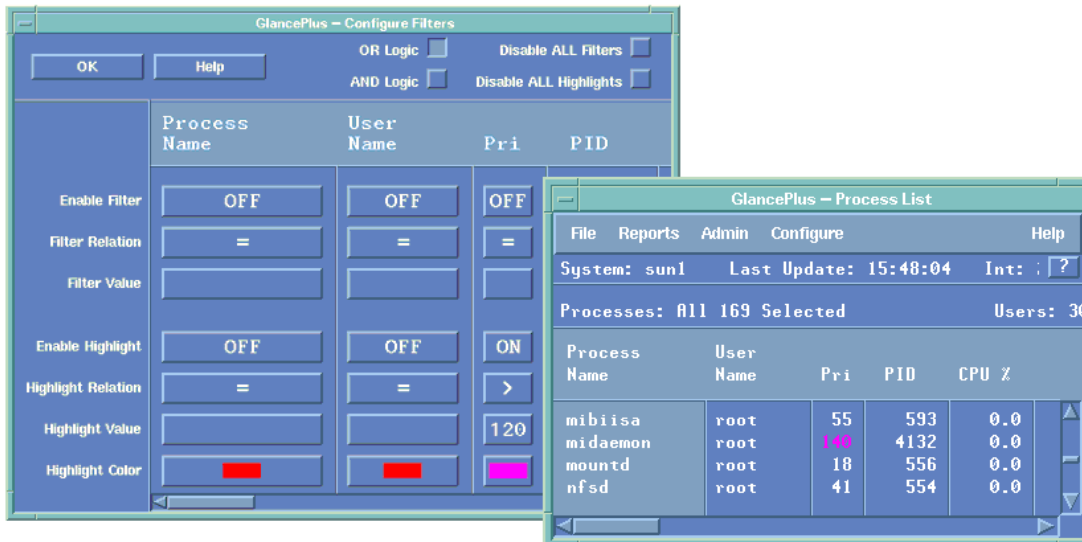
Click AND Logic to display processes in the Process List window that match all of the criteria you specify.

Click Disable ALL Filters to turn off all process filters.

NOTE: If you are filtering and you have a process already highlighted in the Process List window, that process continues to display even if you have set the filters to exclude it.

Highlight Metrics

To specify a metric to highlight:



Select Filters from the Configure menu in the Process List window.
 Locate the first process metric you want to highlight.
 Click Enable Highlight to turn it on.
 Click the Highlight Relation button until the relation symbol you want is displayed.
 Click the Highlight Value button, then type a value.
 Click the Highlight Color button until you get the desired highlight color.
 Click OK when you are finished.

Click Disable ALL Highlights to turn off all metric highlights.

Modifying Runtime Parameters

GlancePlus uses a defaults file, called Gpm, to control certain run-time parameters. The application defaults for GlancePlus are stored in:

`/var/opt/perf/app-defaults/C/Gpm`

You can edit this file to:

- change the system-wide behavior of GlancePlus
- override the application defaults in your local X server database

Before you edit the file, save a copy in case you need to restore the original.

The parameters you can change are:

[Font Type and Size](#)

[Icon Size](#)

[Column Width](#)

If you just want to define these parameters for your own display, make the changes in your local .Xdefaults file.

Font Type and Size

You can override the default font types and sizes used in GlancePlus. Use the following lines to change the font type or size:

```
GpmtinyFont
GpmsmallFont
GpmmediumFont
GpmlargeFont
GpmvariableTextFont
GpmwidgetButtonFont
```

The Gpm*widgetButton Font is the font used in the menus. The Gpm*variableTextFont is used for graph labels. See the **Gpm** file for an example of each font's format.

Icon Size

If your GlancePlus icon is too large or too small, you can override the default icon size. Use the following two lines to change the icon size:

```
GpmiconWindowHeight:nn
GpmiconWindowWidth:nn
```

Be aware, however, that GlancePlus only gives you the largest icon size permitted by your display.

Column Width

You can configure the width of certain columns that contain highly variable data widths. The default and maximum column width vary for different types of columns. Column names that exceed the current width settings are truncated to fit the column. As explained in the truncation rules, names may be truncated from the left, in the center, or on the right, depending on the metric being represented.

NOTE: You can enable the tooltip feature to display the entire length of a truncated string. Refer to Display Truncated String Data for more information.

Column width is controlled by various X resources provided as commented lines in the `apps-defaults` file, located in:

```
/var/opt/perf/app-defaults/C/Gpm
```

To change a column width for all users, uncomment the appropriate resource in the `app-defaults` file and set the column width as desired, within allowable ranges. For example, use `Gpm*pathNameWidth` to change the width of the column for device names, directory names, swap space names, and logical volume names.

After the change, users entering `xglance` see the affected columns at the reconfigured width. Resources are provided for path names, file names, transaction name, and logical volume group name.

To implement these changes for individual users, set these X resources locally, as explained in the man page for X.

Resource names, default width, minimum/maximum width, applicable metrics, and truncation rules are as shown in the table.

Resources Table

Controlling Resource	Default Width	Min/Max Range	Metric Name	Truncation Rules
Gpm*pathNameWidth	20	15-40	BYDSK_DEVNAME	center
			BYDSK_DIRNAME	center
			FS_DEVNAME	left
			FS_DIRNAME	left
			LV_DIRNAME	left
			BYSWP_SWAP_SPACE_NAME	left
Gpm*openFileNameWidth	32	15-68	PROC_FILE_NAME	center
			PROC_REGION_FILENAME	center
Gpm*transactionNameWidth	20	15-128	TT_NAME	center
			TT_USER_MEASUREMENT_NAME	center
			TT_CLIENT_TRAN_NAME	center
Gpm*transactionInfoWidth	20	15-128	TT_INFO	center
Gpm*transactionAppNameWidth	20	15-128	TT_APP_NAME	center
			TT_CLIENT_APP_NAME	center
Gpm*transactionUserNameWidth	15	15-128	TT_UNAME	center
			TT_CLIENT_USER_NAME	center
Gpm*lvGroupNameWidth	11	10-20	LV_GROUP_NAME	right

Version Window Display Time

You can set the number of seconds that the GlancePlus version window displays at startup. Use `Gpm*aboutWindowTime=n` where `n` is the value in seconds. A value of 0 seconds will cause the window not to display at all.

For example:

Gpm*aboutWindowTime=0 will give no display

Gpm*aboutWindowTime=5 will display the Version window for 5 seconds.

Truncation Rules

Names that are truncated on the left use * (asterisk) to replace the missing characters. The following names are truncated on the left:

- file system device names
- directory names
- swap space names
- logical volume names

Names that are truncated in the middle use . . . (three dots) to replace the missing characters. The following names are truncated in the middle:

- process file names
- disk device names
- directory names
- transaction names
- transaction application names
- transaction user names
- transaction information

Names that are truncated on the right have no indicator of the missing characters. The following names are truncated on the right:

- • logical volume group names

NOTE: You can enable the tooltip feature to display the entire length of a truncated string. Refer to Display Truncated String Data for more information.

Display Truncated String Data

GlancePlus tooltips allow you to see the full text of string data that has been truncated in a row-column report window. Tooltips are available in any GlancePlus window that displays data in a row-column format; they are not available in graph windows. The text of a tooltip automatically updates as the data changes. Tooltips can be turned on and off; they are turned on by default.

There are two ways to enable/disable tooltips:

- 1 Select the Configure menu from the top of the report window, then click **Tooltips**.
- 2 Click the "T" button in the top right corner of the window.

To display the tooltip:

- make sure tooltips are enabled,
- move the cursor over the truncated string and hold it there for about 1 second.

NOTE: When tooltips are turned on, the **Tooltips** item on the Configure menu displays **Tooltips (on)** and the T button in the top right corner of the window appears to be pressed in (it's a different color).

Speed Keys

You can navigate using the following speed keys as an alternative to using the mouse.

Speed Key	Action
Alt + underlined character	Open the menu. For example, Alt+F opens the <u>F</u> ile menu.
Arrow keys	Navigate up and down through opened submenus. Open a

selected submenu by pressing **[Enter]**.

<Ctrl>F	Find in the Process List Window. Opens the Search List Dialog Window to search for a process by Process Name or by Process Command Line.
<Ctrl>M	Raise Main Window -- Bring the GlancePlus Main Window to the front
<Ctrl>N	Find Next in the Process List Window. Search for each instance in the list.
<Ctrl>Q	Exit GlancePlus
<Ctrl>U	Update Now -- Update the metrics on the screen.
<Ctrl>W	Close Window -- Close the active window
<Ctrl>Z	Reset Cum to Zero -- Reset cumulative values to zero

Defining Applications

To define applications, you edit the `parm` file, an ASCII text file, located in this directory:

```
/var/opt/perf
```

GlancePlus uses the `parm` file to define applications. You can group related processes together into an application to monitor the combined effect of those processes on such things as disk, memory, and CPU activity. Grouping your processes in this way lets you quickly tune your system for maximum performance.

In GlancePlus, the Application List and Application History windows display metrics for applications defined on the system you're monitoring.

[Creating a Personal Parm File](#)

[Changing the Parm File](#)

[Parm File Syntax](#)

Create a Personal Parm File

If you want to add or change applications, or make any other changes to the `parm` file (such as changing the system ID) without affecting other performance products or users, you can use a personal `parm` file.

Simply copy the `parm` file to your home directory and rename it `.parm`. Then make any necessary changes. GlancePlus always looks for a personal `.parm` file first.

Change the Parm File

You can use almost any text editor that produces standard ASCII files to create or modify the `parm` file. The result must be a standard ASCII file. When you create or modify the `parm` file, the following rules and conventions apply: You need only specify a parameter if you want to override a default.

The order in which the parameters are entered into the `parm` file is not important except as follows:

- If a parameter is entered more than once, the last one entered is used.
- The `file`, `group`, `priority`, `argv1`, `cmd`, `or` and `user` statements must follow the application statement that they define.

You can use uppercase letters, lowercase letters, or a combination of both for all commands and parameter statements.

You can use blanks or any other non-alphanumeric characters (such as semicolons and commas) to separate key words in each statement. You can comment the `parm` file. Any line starting with a comment code (*) or pound sign (#) is ignored.

Modify the Parm File

To modify the `parm` file:

1. Make a backup copy of the `parm` file.
2. Edit the `parm` file on any system. If you change existing application definitions in the `parm` file, GlancePlus running on that system is affected.

3. Save the file.
4. Restart GlancePlus.

Parm File Syntax

ID =ID name
javaarg =true/false
Application=application name
argv1 =first command argument [,]
cmd =command line regular expression
File =list of files
User =list of user names
Group =list of user group names
Or =or
Priority =range

System ID

The system ID value is a string of characters that identifies your system. If you have multiple systems, use different ID strings on each one.

You can specify a maximum of 40 characters.

The default ID is the nodename as returned by `uname -n`.

Log

Specifying `global` in the `log` parameter causes global records to be written to the `logglob` log file.

You must log global data in all cases. You must have global data records to view and analyze performance data on your system.

Specifying `application` in the `log` parameter causes application records to be written to the `logappl` log file.

Specifying `process` in the `log` parameter causes interesting processes to be written to the `logproc` log file. (A process becomes interesting when it is first created, when it terminates, or when it exceeds certain user-defined thresholds.)

Specifying `device=disk` in the `log` parameter causes individual disk, volume, or LAN device records to be written to the `logdev` file.

The default is `log=global,process,device=disk`

All of the log files are created automatically if logging to them is specified and they do not already exist. If a particular type of logging is disabled, the corresponding log file is not removed.

NOTE:

If you specify `log` without options, the default global, process, and disk device data are logged.

Threshold

You can change interesting process thresholds by changing their values. The `cpu` option sets the percentage of CPU utilization that a process must exceed to become *interesting* and be logged. It is used only if process logging is enabled.

The value *percent* is a real number indicating overall CPU use. For example, `cpu=7.5` indicates that a process is logged if it exceeds 7.5 percent of CPU utilization in a 1-minute sample.

The `disk` option sets the rate of major page faults or block IOs that a process must exceed to become interesting and be logged. The value *rate* is the disk IO rate in transfers per second, and is a real number. For example, `disk=8.0` indicates that a process will be logged if it exceeds either 8 major page faults or 8 block IOs per second in a 1-minute sample.

You can enter the statements for thresholds on the same parameter line (separated by commas) or on separate lines.

The default for threshold is:

```
threshold cpu=10.0,disk=10.0
```

The `nonew` option turns off logging of new processes if they have not exceeded any threshold.

The `nokilled` option turns off logging of exited processes if they did not exceed any threshold.

Size

Use the `size` parameter to set the maximum size in megabytes of any raw log file. By default, `logglob` (the global file), `logappl` (the application file), and `logdev` (the device file) have a maximum size of 10 megabytes each. The process file `logproc` has a maximum size of 20 megabytes. You cannot set the size to be less than one megabyte.

The `scopeux` collector reads these specifications when it is initiated from `scope.start`. If any of these log files achieve their maximum size during collection, they are rolled back dynamically at the next maintenance time. During the rollback, the oldest 25 percent of the data is removed from the logfile.

If the `size` specification in the `parm` file is changed, `scopeux` detects it during startup and takes appropriate action. If the maximum log file size is decreased to the point where existing data does not fit, an automatic resize takes place. If the existing data fits within the new maximum size specified, no action is taken.

Application

The application name defines an application or class that groups together multiple processes and reports on their combined activities. The name is a string of up to 19 characters used to identify the application in the `parm` file.

```
application=application name
```

See the application parameter rules for additional details.

Application Parameter Rules

- The `parm` file is processed in the order entered and the first match of program name, group, and/or user `login` defines the application to which a particular process belongs.
- The `application` statement must precede any combination of `file`, `user`, `group`, `cmd`, **or**, `argv1`, or `priority` statements that refer to it, with all such statements applying against the last `application` workload definition.
- Each statement can be up to <nobreak"170 characters" long including the carriage return, with no continuation characters permitted. If your list of files, users, or groups is longer than 170 characters, continue the list on the next line after another `file`, `user`, or `group`.
- If `user`, `file`, `group`, `cmd`, `argv1`, or, and `priority` parameters are specified for the same application, a process must match one of the file name type parameters. The user login must match one of the `user` and `group` parameters and fall within the specified priority range in order to belong to a particular application. A process cannot belong to a particular application if it fails to match any of the four parameters.
- You can define up to 31 applications as needed. A predefined application called "other" collects all processes that are not defined by application parameters in the `parm` file.
- Any process on the system belongs to one application only. Processes are based on name, not program path. Therefore, two processes with the same name but different paths (file system location), are considered to be the same process.
- If a program file (such as a process) is included in more than one application, the process belongs to the first application that contains it.

For example:

```
application=Prog Dev
file=vi,cc,ccom,pc,pascomp,dbx,xdb
application=xyz
file=xyz*,startxyz
```

You can have a maximum of <nobreak"300 file", <nobreak"300 user", and <nobreak"300 group" specifications for all applications combined. The previous example includes nine file specifications.

(`xyz*` counts as only one specification even though it can match more than one program file.)

By default, no user applications are defined.

File

The `file` parameter specifies which program files belong to an application. All interactive or background executions of these programs are included. The parameter applies to the last `application` statement issued. An error is generated if no `application` statement is found.

The `file name` can be any of the following:

- a single UNIX program file such as `vi`
- a group of UNIX program files (indicated with a wildcard) such as `xyz`

In this case, any program name that starts with the letters `xyz` is included. A file specification with wildcards counts as only one specification toward the maximum of 300 for all `file`, `user`, and `group` specifications.

You can enter multiple file names on the same parameter line (separated by commas) or in separate `file` statements. File names cannot be qualified by a path name.

For example:

```
application=payroll
file=account1,basepay,endreport
application=Prog Dev
file=vi,cc,ccom,pc
file=pascomp,xdb
```

If you do not specify a `file` parameter, all programs that satisfy the other parameters qualify.

User

The **user** parameter specifies which user names belong to an application.

For example:

```
application=Prog Dev Group 2
file=vi,xb,abb,ld,lint
user=ted,rebecca,test*
```

User specifications that include wildcards count as only one specification toward the maximum of 300 for all file, user, and group specifications.

If you do not specify a `user` parameter, all programs that satisfy the other parameters qualify.

Group

The **group** parameter specifies which user group names belong to an application.

For example:

```
application=Prog Dev Group 2
file=vi,xb,abb,ld,lint
user=ted,rebecca,test*
group=lab,test
```

If you do not specify a `group`, all programs that satisfy the other parameters qualify.

Or

Use the `or` parameter to allow more than one application definition to apply to the same application. Within a single application definition, a process must match at least one of each category of parameters. Parameters separated by the `or` parameter are treated as independent definitions. If a process matches the conditions for any definition, it belongs to the application. For example:

```
application=Term12
user=sysop

or

user=sysman
file=vi,store,dmp
```

This defines the application (`Term12`) that consists of any programs run by the user `sysop` plus other programs (`vi,store,dmp`) if they are executed by the user `sysman`.

Priority

The priority of processes can range from 0 to 255 for the standard HPUX scheduler. (Other schedulers have different priority ranges.) The list below specifies the range reserved for each process mode.

\0-127\Real Time

\128-177\System

\178-255\Time Share (User)

You can restrict processes in an application to those belonging to a specified range by using the `priority` parameter. For example:

```
application=swapping
priority=128-131
```

This parameter groups all processors running at PSWP priority into one application. If you do not specify a priority, all programs that satisfy the other parameters qualify.

The priority of a process can change over the life of the process. The scheduler adjusts the priority of time-share processes. A user can also change priority programmatically or while the process is executing, with the `nice` and `rtprio` commands.

The process priority is sampled at the end of each one-minute sample interval. If the process has changed priority, it can change applications. All activity for a process during the one-minute interval is assumed to have occurred at the new priority and is attributed to the application that matches the process at the end of each one-minute sample interval.

Application Definition Examples

The following list shows application definition examples.

```
application=Real Time
priority=0-127
application=Prog Dev Group 1
file=vi,dbx,abb,ld,lint
user=bill,debbie
application=Prog Dev Group 2
file=vi,dbx,abb,ld,lint
user=ted,rebecca,test*
group=labmqa
application=Other Editors
file=ed,sed,awk
application=Compilers
file=cc,ccom,xlc,c++fe
application=Users
user=nelson,ted,rebecca,gene
```

cmd

The `cmd` parameter specifies processes for inclusion in an application by their command strings, which consists of the program executed and its arguments (parameters). Unlike other selection parameters, this parameter allows extensive wildcarding besides the use of the asterisk character.

Similar to regular expressions, extensive pattern matching is allowed. For a complete description of the pattern criteria, see the UNIX man page for `fnmatch`. Unlike other parameters, you can have only one selection per line, however you can have multiple lines.

The following shows use of the `cmd` parameter:

```
application = newbie
cmd = *java *[Hh]ello[Ww]orld*
```

javaarg

The `javaarg` parameter is a flag that can be set to `true` or `false`. It **ONLY** affects the value of the `proc_proc_argv1` metric.

When `javaarg` is set to `false` or is not defined in the `parm` file, the `proc_proc_argv1` metric is always set to the value of the first argument in the command string for the process.

When `javaarg` is set to `true`, the `proc_proc_argv1` metric is overridden, for java processes only, with the `class` or `jar` specification if that can be found in the command string. In other words, for processes whose file names are `java` or `jre`, the `proc_proc_argv1` metric is overridden with the first argument without a leading dash not following a `-classpath` or a `-cp`, assuming the data can be found in the argument list provided by the OS.

While this sounds complex, it is very plain when you have java processes running on your system: set `javaarg=true` and the `proc_proc_argv1` metric is logged with the `class` or `jar` name. This can be very useful if you want to define applications specific to java. When the class name is in `proc_proc_argv1`, then you can use the `argv1=` application qualifier (explained below) to define your application by class name.

argv1

The `argv1` parameter specifies which processes are selected for the application by the value of the `PROC_PROC_ARGV1` metric. This is normally the first argument of the command line, except when `javaarg=true`,

when this is the class or jar name for java processes. This parameter uses the same pattern matching syntax used by parm parameters like file= and user=. Each selection criteria can have asterisks as a wildcard match character, and you can have more than one selection on one line separated by commas.

For example, the following application definition buckets all processes whose first argument in the command line is either -title, -fn, or -display:

```
application = xapps
argv1 = -title,-fn,-display
```

The following application definition buckets a specific java application (when javaarg=true):

```
application = JavaCollector
argv1 = com.*Collector
```

The following shows how the argv1 parameter can be combined with the file parameter:

```
application = sun-java
file = java
argv1 = com.sun*
```

Introduction to the GlancePlus Adviser

The GlancePlus Adviser monitors your system; it looks for performance metrics that are exceeding their defined thresholds and notifies you when such a condition exists. It sends alarms when specified conditions occur, and notifies you of symptoms of potential bottlenecks.

The Adviser gets its commands from a text file that you can customize to suit the needs of your organization. You can modify the syntax of the Adviser text file to define performance metric thresholds, such as:

- when global swap space is nearly full,
- when the system process table is near capacity, and
- when your CPU has been running at 90% busy for more than 2 minutes.

The Adviser notifies you when it detects a condition that exceeds the specified thresholds. You can configure it to:

- display information to stdout,
- execute a UNIX mail command, such as mailx, to send a message,
- make the GlancePlus ALARM button turn yellow or red or, if you are running GlancePlus iconified, it can place a red or yellow border around the GlancePlus icon, or
- display a specific GlancePlus window to help you analyze the problem.

You specify Adviser symptoms and alarms in the syntax of the Adviser text file. The syntax defines each of the specific thresholds and rules as well as the actions that are triggered if certain conditions are present.

The Adviser syntax to be used is specified in a file that is identified at run time with the -syntax <filenameoption. If no syntax file is specified, the Adviser looks for a user default file named adviser.syntax in your home directory. If no user default is found, the Adviser looks for a system default syntax file named adviser.syntax in the /var/opt/perf/ directory.

By default, the GlancePlus Adviser is turned on whenever you run **glance** or **xglance**. If you like, you can turn it off by using the -adviser_off run-time parameter when you start **glance** or **xglance**.

Any output produced by the Adviser is sent to the file adviser.out in your local directory.

You can also specify that the Adviser run alone without the GlancePlus user interface. In this mode, Adviser sends its messages to stdout. To run GlancePlus and the Adviser in this way, include the -adviser_only option when you start **glance** or **xglance**.

A good way to learn how to customize the Adviser syntax is to make small modifications to the default Adviser syntax file. The default Adviser file is /var/opt/perf/adviser.syntax.

Choose one of the topics below to learn more about the Adviser:

[Adviser Syntax Structure](#)

[Adviser Syntax Reference](#)

[Editing Adviser Syntax](#)

Alarms and Symptoms

Alarms are simply a way to highlight metric conditions in GlancePlus. A symptom is a combination of conditions that occurs during an interval and contributes to a bottleneck on your system.

What is an Alarm?

An alarm can trigger whenever conditions that you specify are met. Alarms are based on any period of time you specify, which can be one interval or longer. Conditions or events that you might want to set as Adviser alarms include:

- when global swap space is nearly full
- when the page in rate is too high
- when your process table is near capacity
- when your CPU has been running at 75% utilization for the last two minutes

Several screens let you look at alarm status and history. The status of alarm conditions determines the color of the main window's Alarm button. Several alarms are defined in the GlancePlus default Adviser syntax. (To see the default syntax, open the Edit Adviser Syntax window in GlancePlus.)

What is a Symptom?

Complex alarms can be built based on symptoms. The GlancePlus default Adviser syntax defines four bottleneck symptoms for you, then defines alarms based on those symptoms. (Open the Edit Adviser Syntax window in GlancePlus to see the default syntax.)

By observing different metrics with corresponding thresholds and adding values to the probability that these metrics contribute to a bottleneck, the Adviser calculates one value that represents the combined probability that a bottleneck is present.

Unlike the ALARM statement that monitors conditions over a period of time normally longer than one interval, the SYMPTOM statement is evaluated and updated every interval. This is why you might see the CPU Bottleneck Symptom indication prior to a CPU Bottleneck Alarm. Symptoms change rapidly and can become yellow, then red, then go back to green. An alarm remains yellow or red until it is reviewed or reset.

You can also use the variables you defined in the SYMPTOM statements in the Alarm section. And you can link the symptoms to the CPU, Disk, Network, and Memory buttons on the main GlancePlus window to notify you of possible bottlenecks.

For every symptom that you define in the Adviser Syntax window, a graph appears on the Symptom History window to show that particular symptom's probability over time.

Editing Adviser Syntax

Don't worry too much about making mistakes; you can always go back to the default Adviser syntax by selecting the Default Syntax option from the Reset menu in the Edit Adviser Syntax Window.

You can edit the syntax in two ways:

- [Using the GlancePlus Text Editor](#)
- [Using Your Own Text Editor](#)

Using the GlancePlus Text Editor

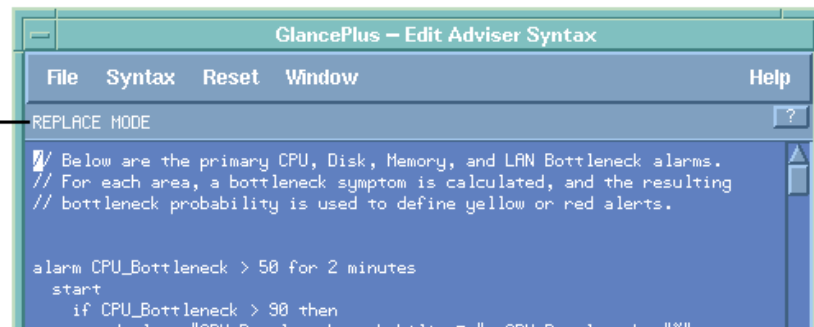
You can edit the adviser syntax from within GlancePlus. Here's how you do it.

Open the Edit Adviser Syntax window from the Adviser menu on the Main window.

Select either the Alarm window (default) or the Symptom window using the Window menu.

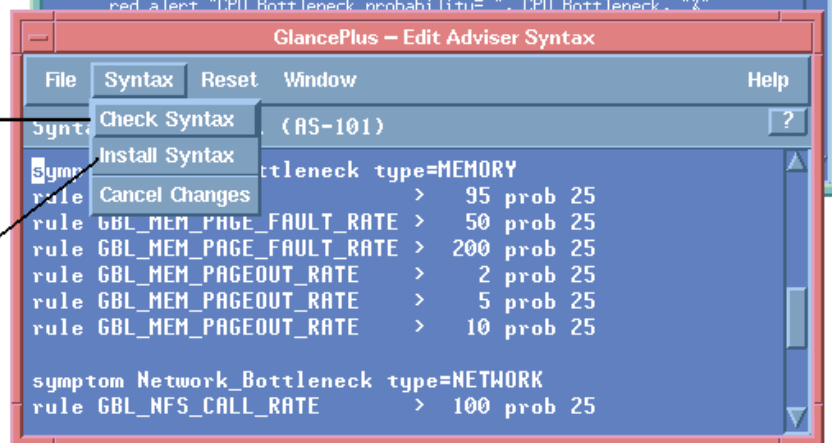
Alarm Window

The text editor defaults to Replace Mode. For information on all the editing commands, see below.



Make your edits, then select Check Syntax from the Syntax menu.

If everything checks OK, then select Install Syntax from the Syntax menu.



Symptom Window

Syntax Editing Commands

To edit text in the Adviser Syntax Window, you use various editing commands. You cannot use the mouse to move the cursor in the Adviser Syntax window.

To replace text:

Each time you open the Adviser Syntax window, the editing function is in REPLACE MODE. You can overwrite the syntax with characters or blanks using the Replace Mode.

To return to REPLACE MODE after inserting text, press the `Insert char` key.

To insert text:

To insert information in the Adviser Syntax window, press the `Insert char` key. The message at the top of the Adviser Syntax window changes to INSERT MODE.

To insert lines or characters, use the `Insert line` or `Insert char` key.

To delete text:

To delete lines or characters, use the `Delete line` key or the `Delete char` key.

Moving the cursor in the Adviser Syntax window:

To move the cursor one character at a time, use your keyboard arrow keys.

To page through text, use the `prev` and `next` keys, or use the vertical scroll bar on the right.

To scroll through text horizontally, use the shift key and the left or right arrow keys.

Using Your Own Text Editor

You can edit the adviser syntax using your favorite text editor. Here's how to do it.

Open the Edit Adviser Syntax window from the Adviser menu on the Main window.

Select either the Alarm window (default) or the Symptom window using the Window menu.

Alarm Window

Select Store to File. Enter the full path and file name you are using.

Using your favorite text editor, make your changes.

Select Load from File. Enter the path and file information.

Select Check Syntax from the Syntax menu. If everything checks OK, then select Install Syntax from the Syntax menu. Your changes are now in effect.

Print CPU Total Utilization

Follow these steps to print metric values to the xglance window:

1. From the Main window, select Edit Adviser Syntax from the Adviser menu to open the Edit Adviser Syntax window.
2. In the Edit Adviser Syntax window, press the Insert (or Insert Char) key and then press Return a few times to insert several blank lines at the top of the file.
3. Insert the following text in the space you just created at the top of the syntax:

```
print gbl_cpu_total_util
```
4. From the Syntax menu, select Install Syntax. The Edit Adviser Syntax window closes and the print statement executes the next time GlancePlus updates its data. When you select Install Syntax, GlancePlus checks your syntax for correctness. If an error is found, an error message is displayed at the top of the window. For an explanation of any syntax error messages, see GlancePlus Messages.
5. Look at the window from which you started GlancePlus. The numbers appearing in that window result from GlancePlus printing the value of a global GlancePlus metric (your global CPU utilization) every update interval.

Printing CPU Utilization During High CPU Usage

Perhaps you want to print CPU utilization only when usage exceeds 90% busy.

1. 1. Go back to the Edit Adviser Syntax Window and replace the line you typed with the following:

```
IF gbl_cpu_total_util 90 THEN  
    print "total cpu utilization is high: ", gbl_cpu_total_util
```
2. 2. From the Syntax menu, select Install Syntax. The Edit Adviser Syntax window closes, and the print statement executes the next time GlancePlus updates its data. When you select Install Syntax, GlancePlus checks your syntax for correctness. If an error is found, an error message is displayed at the top of the window. For an explanation of any syntax error messages, see GlancePlus Messages.
3. 3. Look at the window from which you started GlancePlus. You may not see any numbers because data only displays when your CPU is more than 90% busy.
4. 4. To start a program that uses a lot of CPU and view what happens, type the following at a shell prompt (sh or ksh) to cause a loop:

```
while true  
do
```

```
A=1
done
```

This makes the shell loop until you interrupt it with `control-c`. When the loop starts, the Adviser starts printing out information.

Sending Email Messages

You can use metrics that are shown in different GlancePlus windows in your Adviser syntax. Rather than printing metrics to `stdout`, you can send the same information to yourself in an email message.

1. Go to the Edit Adviser Syntax Window, and replace the line you typed with the following:

```
IF gbl_cpu_total_util 90 THEN
    exec "echo 'cpu is too high' ", gbl_total_util, "% ' | mail root"
```

2. From the Syntax menu, select Install Syntax. The Edit Adviser Syntax window closes.

When you select Install Syntax, GlancePlus checks your syntax for correctness. If an error is found, an error message is displayed at the top of the window. For an explanation of any syntax error messages, see GlancePlus Messages.

Printing Process Information Within a Loop

To customize your syntax further, you can combine metrics, define variables, and use looping constructs. This example shows how you can:

- construct loops inside conditions which only execute when a potential problem situation arises
- use variables inside the adviser syntax to keep track of things inside loops. You could change the thresholds in this example to isolate problems unique to your environment.

This example tests for an overall high global system mode CPU utilization. When GlancePlus encounters this situation, it loops through all the active processes, printing out information about the process with the highest percentage of time spent in system mode.

1. Go back to the Edit Adviser Syntax Window, and replace the line you typed with the following:

```
// check for high system-mode cpu utilization, and when it is high,
// print the highest sys cpu consuming process:
IF gbl_cpu_sys_mode_util 5 THEN {
    highestsys = 0
    process loop
        IF proc_cpu_sys_mode_util highestsys THEN {
            highestpid = proc_proc_id
            highestname = proc_proc_name
            highestsys = proc_cpu_sys_mode_util
        }
    print "--- High system cpu rate = ", gbl_cpu_sys_mode_util, " at ",
        gbl_stattime, " ---"
    print "    Process with highest system cpu was pid ", highestpid " 5 " 0,
        ", name: ", highestname
    print "    which had", highestsys, " percent system mode cpu utilization"
}
```

2. From the Syntax menu, select Install Syntax. The Edit Adviser Syntax window closes, and the print statement executes the next time GlancePlus updates its data.

Print to a File

You can print information to a file by using the PRINT statement in the Adviser Syntax and by rerouting `stdout` to a file.

By using the PRINT statement, which sends its output to the defined `stdout` of GlancePlus, you can format metrics with literal constants and user-defined variables. To reroute the `stdout`, start GlancePlus by appending a `>filename` to the command line. This causes all output destined for `stdout` to be placed in the file specified by `filename`.

The following example shows how to print global and process metrics to a file:

Start GlancePlus at the command line.



```
hpterm
sun1: /usr/caryn>
sun1: /usr/caryn>
sun1: /usr/caryn>
sun1: /usr/caryn>
sun1: /usr/caryn>
sun1: /usr/caryn>
sun1: /usr/caryn>
sun1: /usr/caryn> gpm > myoutputfile
```

Adviser Syntax Structure

The Adviser syntax is a simple script language that allows you to set alarms and define symptom conditions. These alarms and symptoms monitor your system and notify you when selected performance metrics are exceeding threshold limits.

A default syntax file is provided in `/var/opt/perf/adviser.syntax`. You can edit the syntax file to define your own alarms and symptoms.

A specific syntax file can be requested at run time with the `-syntax <filenameparameter>`. If no syntax file is specified, the Adviser looks for a user default file, `adviser.syntax` in your home directory. If no user default is found, the Adviser looks for the system default syntax file in `/var/opt/perf/adviser.syntax`

This section shows the structure for both the Alarm and Symptom syntax. To get more detailed information on the elements of the syntax, click the desired element.

Alarm Syntax

```
ALARM condition      [FOR duration{SECONDS, MINUTES, INTERVALS}]
    [condition [FOR duration{SECONDS, MINUTES, INTERVALS}] ...
    [START statement]
    [REPEAT [EVERY duration {SECONDS, MINUTES, INTERVALS}] statement]
    [END statement]

    [(RED or CRITICAL), (YELLOW or WARNING), RESET] ALERTALERT_Statement printlist
ALIAS variable = " alias name "
[VAR] variable = expression
    {
```

```

        compound statements
    }
EXEC printlist
GPM rpt reportlist
IF condition
    THEN statement
    [ELSE statement]
{APPLICATION, APP, CPU, DISK, FILESYSTEM, FS, LAN, LOGICALVOLUME, LV, NETIF, NFS,
NFS_OP, PROCESS, PROC, PROC_FILE, PROC_REGION, SWAP, TRANSACTION, TT, TTBIN,
TT_CLIENT, TT_INSTANCE, TT_UDM, TT_INSTANCE_CLIENT, TT_INSTANCE_UDM, TT_CLIENT_UDM}
LOOP statement
PRINT printlist

```

Symptom Syntax

```

SYMPTOM variable [ TYPE = {CPU, DISK, MEMORY, NETWORK}]
RULE measurement {>, <, <=, >=, ==, !=} value PROB probability
[RULE measurement {>, <, <=, >=, ==, !=} value PROB probability]

```

For detailed Adviser Syntax reference information, refer to Adviser Syntax Reference.

Adviser Syntax Reference

Click any item below to learn more about the Adviser syntax and how to use it. Each statement definition includes an example. For more complex examples, click [Alarm Examples](#).

Syntax Elements

- [Syntax Conventions](#)
- [Comments](#)
- [Conditions](#)
- [Constants](#)
- [Expressions](#)
- [Metric Names](#)
- [Printlist](#)
- [Variables](#)

Statements

- [ALARM Statement](#)
- [ALERT Statement](#)
- [ALIAS Statement](#)
- [ASSIGNMENT Statement](#)
- [COMPOUND Statement](#)
- [EXEC Statement](#)
- [GPM Statement](#)
- [IF Statement](#)
- [LOOP Statement](#)
- [PRINT Statement](#)
- [SYMPTOM Statement](#)

Syntax Conventions

- Braces ({}) indicate that one of the choices is required.
- Brackets ([]) indicate an optional item.
- Items separated by commas within brackets or braces are options. Choose only one.
- Italics indicate a variable name that you will replace.
- All CAPS are Adviser syntax keywords.

Comments

Syntax:

```
# [any text or characters]
or
// [any text or characters]
```

You can precede comments either by double forward slashes (//) or the pound sign (#). In both cases, the comment ends at the end of the line.

Conditions

A condition is defined as a comparison between two metric names, user variables, or numeric constants.

```
item1 {>, <, >=, <=, ==, !=} item2 [ OR item3 {>, <, >=, <=, ==, !=} item4
or:
item1 {>, <, >=, <=, ==, !=} item2 [ AND item3 {>, <, >=, <=, ==, !=} item4
("==" means "equal", and "!=" means "not equal".)
```

Conditions are used in the ALARM statement and the IF statement. They can be used to compare two numeric metrics, variables or constants, and they can also be used between two string metric names, user variables or string constants. For string conditions, only == or != can be used as operators.

You can use compound conditions by specifying the OR or AND operator between subconditions.

Condition Examples

```
gbl_swap_space_reserved_util 95
proc_proc_name == "test" OR proc_user_name == "tester"
proc_proc_name != "test" AND proc_cpu_sys_mode_util highest_proc_so_far
```

Constants

Constants can be either numeric or alphanumeric. An alphanumeric constant must be enclosed in double quotes. There are two kinds of numeric constants: integer and real. Integer constants may contain only digits and an optional sign indicator. Real constants may also include a decimal point.

Constants are useful in expressions, conditions, and statements. For example, you may want to compare a metric against a constant numeric value inside a condition to generate an alarm if it is too high.

Constant Examples

```
345      Numeric integer
345.2    Numeric real
"Time is" Alphanumeric literal
```

Expressions

Use expressions to evaluate numerical values. An expression can be used in a condition or an action.

An expression can contain:

- numeric constants [Constants>morehelp](#)
- numeric metric names [Metric_Names>morehelp](#)
- numeric variables [Variables>morehelp](#)
- an arithmetic combination of the above
- a combination of the above grouped together using parentheses

Expression Examples

```
Iteration + 1
3.1416
gbl_cpu_total_util - gbl_cpu_user_mode_util
( 100 - gbl_cpu_total_util ) / 100.0
```

Printlist

The printlist is any combination of properly formatted expressions, Metric Names, user variables, or constants. See the examples for the proper formatting.

Printlist Examples

Expressions

```
expression ["width["decimals]]
```

Metric Names or User Variables

```
metric names ["width["decimals]]
```

or

```
user variables ["width["decimals]]
```

The metric names or user variables must be alphanumeric.

Constants

No formatting is necessary for constants.

Formatted Examples

```
gbl_cpu_total_util"6"2    formats as  '100.00'  
(100.32 + 20)"6         formats as  '  120'  
gbl_machine"5           formats as  '7013/'  
"User Label"           formats as  "User Label"
```

Metric Names in Adviser Syntax

You can directly reference metrics anywhere in the Adviser syntax. You can use the following types of metrics in the Adviser syntax:

- global metrics (prefixed with `gbl_` or `tbl_`)
- application metrics (prefixed with `app_`)
- process metrics (prefixed with `proc_`)
- disk metrics (prefixed with `bydisk_`)
- by CPU metrics (prefixed with `bycpu_`)
- file system metrics (prefixed with `fs_`)
- logical volume metrics (prefixed with `lv_`)
- network interface metrics (prefixed with `bynetif_`)
- swap metrics (prefixed with `byswp_`)
- ARM metrics (prefixed with `tt_` or `ttbin_`)

You can use global metric anywhere in the Adviser syntax, but you can only use process, logical volume, disk, file system, LAN, and swap metrics within the context of a LOOP statement.

Metric names can contain alphanumeric (for example, `gbl_machine` or `app_name`) or numeric data and can reflect several different kinds of measurement. For example, the metric ending of a metric name indicates what is being measured:

- `a_util` metric measures utilization in percentages
- `a_rate` metric measures units per second
- `a_queue` metric measures the number of processes or threads waiting for a resource

If you are unsure of the unit of measure for a specific metric, refer to the metric definition in online help.

You must associate an application metric with a specific application, except when using the LOOP statement. To do this, specify the application name followed by a colon, and then the metric name. For example, `other_apps:app_cpu_total_util` specifies the total CPU utilization for the application `other_apps`. Refer to the ALIAS statement description for more information on using application metrics in the syntax.

Application names, as defined by the `parm` file, may contain special characters and embedded blanks. To use these names in the syntax (where application names must match the form of a variable name), the names are made case-

insensitive and embedded blanks are converted to underlines. This means that the application name defined as "Other Apps" may be referenced in the syntax as "other_apps". For application names defined with special characters, you must use the ALIAS statement to specify an alternate name.

When explicitly qualified, application metrics may be referenced anywhere in the syntax. Unqualified application metrics may only be referenced within the context of the LOOP statement. This is an iterative statement which implicitly qualifies application or process metrics.

You can only reference process metrics within the context of a LOOP statement. There is no way to explicitly reference a process.

Variables

Variables must begin with a letter and can include letters, digits, and the underscore character. Variables are not case-sensitive.

The following example defines the numeric variable `highest_CPU_value` by assigning it a value of zero.

```
highest_CPU_value = 0
```

The following example defines the alphanumeric variable `my_name` by assigning it a null string value.

```
my_name = ""
```

ALARM Statement

Use the ALARM statement to notify you when certain events, which you define, occur on your system. Using the ALARM statement, the Adviser can notify you in a number of different ways:

- through messages to the Alarm History window
- through messages sent to your originating shell
- by automatically opening a GlancePlus window

Syntax:

```
ALARM condition [FOR duration {SECONDS, MINUTES, INTERVALS}]
    [condition [FOR duration {SECONDS, MINUTES, INTERVALS}] ] ...
[START statement]
[REPEAT [EVERY duration [SECONDS, MINUTES, INTERVAL, INTERVALS]] statement]
[END statement]
```

The ALARM statement must be a top-level statement. It cannot be nested within any other statement.

However, you can include several ALARM conditions in a single ALARM statement, in which case all conditions must be true for the alarm to trigger. And you can also use a COMPOUND Statement, which is executed at the appropriate time during the alarm cycle.

START, REPEAT, and END are ALARM statement keywords. Each of these keywords specifies a *statement*. You must have a START, REPEAT, or END in an ALARM statement, and they must be listed in correct order.

The alarm cycle begins on the first interval that all of the alarm conditions have been true for at least the specified duration. At that time, the Adviser executes the START *statement*, and on each subsequent interval checks the REPEAT condition. If enough time has transpired, the *statement* for the REPEAT clause is executed. This continues until one or more of the alarm conditions becomes false. This completes the alarm cycle and the END *statement* is executed.

If you omit the EVERY specification from the REPEAT statement, the Adviser executes the REPEAT statement at each interval.

ALARM Examples

The following list contains examples of the ALARM syntax.

- ALARM Example: Typical ALARM Statement
- ALARM Example: Using COMPOUND Statements
- ALARM Example: Using Multiple Conditions
- ALARM Example: Process Table
- ALARM Example: Swap Space
- ALARM Example: CPU Problem
- ALARM Example: Yellow Alert

ALARM Example: Typical ALARM Statement

The following ALARM example sets a red alert when the semaphore table is almost full. It is similar to a predefined Alarm in the default syntax. Do not add this to your syntax without removing the default, or your subsequent alert messages may be confusing.

```
ALARM tbl_sem_table_util 90 FOR 1 MINUTE
    START RED ALERT "Semaphore Table is nearly full"
        REPEAT EVERY 30 SECONDS RED ALERT "Semaphore Table still nearly full"
    END RESET ALERT "End of Semaphore Table full condition"
```

This ALARM example tests the metric `tbl_sem_table_util` to see if it is greater than 90. If it is, the RED ALERT statement changes the ALARM button label on the Main window (or on the GlancePlus icon if you are running in iconified mode) to red and places the "Semaphore Table is nearly full" message in the Alarm History window.

The REPEAT statement checks for the `tbl_sem_table_util` condition every 30 seconds. As long as the condition is greater than 90, the REPEAT tells the Adviser to maintain a RED ALERT condition and sends the "Semaphore Table still nearly full" message to the Alarm History window.

When the `tbl_sem_table_util` condition goes below 90, the RESET ALERT statement turns off the alert color and logs the "End of Semaphore Table full condition" message in the Alarm History window.

ALARM Example: Using COMPOUND Statements

Use the following example to use a COMPOUND statement within the ALARM statement. This example shows you how to make the Adviser open a window when an event occurs and how to print a statement to your originating GlancePlus shell:

```
ALARM cpu_bottleneck 90 FOR 1 MINUTE
    START {
        RED ALERT "Your CPU is bottlenecked."
        GPM rpt cpugraph
        PRINT "CPU is running at: ", gbl_cpu_total_util
    }
    END
    RESET ALERT "CPU crisis is over."
```

ALARM Example: Using Multiple Conditions

You can have more than one test condition in the ALARM statement. In this case, each statement must be true for the alarm button to activate. For example:

```
ALARM gbl_cpu_total_util 90 FOR 2 MINUTES
    gbl_cpu_sys_mode_util 50 FOR 1 MINUTES
    START RED ALERT "The CPU is busy and System Mode CPU utilization is high."
    END RESET ALERT "The CPU alert is now over."
```

This ALARM example tests the metric `gbl_cpu_total_util` and `CPU_Bottleneck`. If both conditions are true, the RED ALERT statement sets a critical alert. When either test condition becomes false, the RESET is executed.

Alarm Example: Process Table

```
ALARM tbl_proc_table_util 90 FOR 1 MINUTES
    START RED ALERT "Proc table is nearly full"
    END RESET ALERT "End of Proc table full condition"
```

This alarm turns the Alarm button red when the process table is full. This red alert alarm also shows up in the Alarm History window.

Alarm Example: Swap Space

```
//GLOBAL SWAP ALARM
symp_swap_util = gbl_swap_space_used / gbl_swap_space_avail
ALARM symp_swap_util 0.9
    START
        RED ALERT "GLOBAL SWAP space is nearly full"
    END RESET ALERT "GLOBAL SWAP space crisis is over"
```

This example shows computing a new variable, `symp_swap_util`, which represents swap utilization. The Adviser will send an alarm when the swap utilization exceeds 90%. On the next interval that `symp_swap_util` falls below 90%, the alarm condition becomes false, and the ALARM is reset.

Alarm Example: Yellow Alert

```
ALARM Symp_Global_Cpu_Bottleneck 50 FOR 2 MINUTES
START
    YELLOW ALERT "CPU Bottleneck probability= ",
        Symp_Global_Cpu_Bottleneck, "% for the last 2 minutes"
REPEAT every 2 minutes
    YELLOW ALERT "CPU Bottleneck probability= ",
        Symp_Global_Cpu_Bottleneck, "% for the last 2 minutes"
END
RESET ALERT " CPU Bottleneck Yellow Alert over, probability=",
    Symp_Global_Cpu_Bottleneck, "%"
```

The ALARM tests the SYMPTOM variable, which is defined in the SYMPTOM Statement `Symp_Global_Cpu_Bottleneck`. If the SYMPTOM variable is greater than 50 for 2 minutes the ALARM notifies you with a YELLOW ALERT to your main GlancePlus window. The CPU Bottleneck probability message is recorded in the Alarm History window.

The ALARM REPEATs every 2 minutes until the ALARM condition is false. At that time, END RESETs the ALERT and posts the corresponding message to the Alarm History window. During each interval that the `Symp_Global_Cpu_Bottleneck` is greater than 50%, the CPU Util global bar heading is highlighted.

Alarm Example: CPU Problem

```
ALARM
    gbl_cpu_total_util 90 FOR 30 SECONDS
    gbl_run_queue 3 FOR 30 SECONDS
START YELLOW ALERT "CPU AT ", gbl_cpu_total_util, "% at ", gbl_stattime
REPEAT EVERY 300 SECONDS {RED ALERT "CPU AT ", gbl_cpu_total_util
exec "/usr/bin/pager -n 555-3456"}
END ALERT "CPU at ", gbl_cpu_total_util, "% at ", gbl_stattime, " - RELAX"
```

This example lights a yellow alert on the ALARM button or icon and writes a message to the Alarm History window whenever CPU utilization exceeds 90% for 30 seconds and the CPU run queue exceeds 3 for 30 seconds.

If both conditions remain true, xglance generates a red alert, writes another message to the Alarm History window and runs a program to page the system administrator.

When either one of the alarm conditions fails to be true, the ALARM BUTTON or icon resumes its normal color and a message is written to the Alarm History window giving the global CPU utilization, the time the alert ended, and a note to RELAX.

ALERT Statement

The ALERT statement is used to place a message in the Alarm History Window. Whenever an ALARM detects a problem it can execute an ALERT statement to activate the ALARM button label on the Main window or the icon border to notify you of a problem. A user-customized message, specified by *printlist*, records the event in the Alarm History window. You can use the ALERT statement in conjunction with an ALARM statement.

Syntax:

```
[(RED or CRITICAL), (YELLOW or WARNING), RESET] ALERT printlist
```

- RED and YELLOW, are synonymous with CRITICAL and WARNING. These keywords place the *printlist* in the Alarm History window, along with the time and alarm level, in red or yellow characters. They also change the text color of the ALARM button on the Main window to red or yellow, or if iconified, set the icon border to a flashing red or yellow color. If you prefer, you can set a no priority alert (not red or yellow, just information to the Alarm History Window).
- RESET records the *printlist* in the Alarm History window and resets any colors on the icon or ALARM button to their normal color.

See ALERT Example

ALERT Example

An example an ALERT statement is:

```
RED ALERT "CPU utilization = ", gbl_cpu_total_util, " at ", gbl_stattime
```

When executed this statement turns the ALARM button label red or, if GlancePlus is iconified, puts a flashing red border in the icon and writes a message in the Alarm History window that reads, for example:

```
CPU utilization = 85.6 at 14:43:10
```

ALIAS Statement

Use the ALIAS statement to assign a variable to an application name that contains special characters or imbedded blanks.

Syntax:

```
ALIAS variable = "alias name"
```

ALIAS Example

Because you cannot use special characters or imbedded blanks in the syntax, using the application name "other user root" in the PRINT statement below would have caused an error. Using ALIAS, you can still use "other user root" or other strings with blanks and special characters within the syntax.

```
ALIAS otherapp = "other user root"
PRINT "CPU for other root login processes is: ", otherapp:app_cpu_total_util
```

ASSIGNMENT Statement

Use the ASSIGNMENT statement to assign a numeric or alphanumeric value, *expression*, to the user variable.

Syntax:

```
[VAR] variable = expression
[VAR] variable = alphaitem
[VAR] variable = alphaitem
```

ASSIGNMENT Examples

A user variable is determined to be numeric or alphanumeric at the first assignment. You cannot mix variables of different types in an assignment statement.

1. This example assigns an alphanumeric application name to a new user variable:

```
myapp_name = other:app_name
```

2. This example is incorrect because it assigns a numeric value to a user variable that was previously defined as alphanumeric (in example 1):

```
myapp_name = 14
```

3. This example assigns a numeric value to a new user variable:

```
highest_cpu = gbl_cpu_total_util
```

4. This example is incorrect because it assigns an alphanumeric literal to a user variable that was previously defined as numeric (in example 3):

```
highest_cpu = "Time is"
```

COMPOUND Statement

Use the COMPOUND statement with the IF Statement, the LOOP Statement, and the START, REPEAT, and END clauses of the ALARM Statement. By using a COMPOUND statement, a list of statements can be executed.

Syntax:

```
{
  statement
  statement
}
```

Construct compound statements by grouping a list of statements inside braces ({}). The compound statement can then be treated as a single statement within the syntax.

Compound statements cannot include ALARM and SYMPTOM statements. (Compound is a type of statement and not a keyword.)

COMPOUND Example

```
highest_cpu = highest_cpu
IF gbl_cpu_total_util highest_cpu THEN
  // Begin compound statement
  {
    highest_cpu = gbl_cpu_total_util
    PRINT "Our new high CPU value is ", highest_cpu, "%"
  }
  // End compound statement
```

In this example, `highest_cpu = highest_cpu` defines a variable called `highest_cpu`. The Adviser saves the `highest_cpu` value and notifies you only when that `highest_cpu` value is exceeded by a higher `highest_cpu` value.

In the example, if you replaced `highest_cpu = highest_cpu` with `highest_cpu = 0`, then the `highest_cpu` value would be reset to zero at each interval.

You would be notified at each interval what your `highest_cpu` value is.

EXEC Statement

Use the EXEC statement to execute a UNIX command from within your Adviser syntax. You could use the EXEC command, for example, if you wanted to send a mail message to the MIS staff each time a certain condition is met.

Syntax:

```
EXEC printlist
```

The resulting *printlist* is submitted to your operating system for execution.

Because the EXEC command you specify may execute once every update interval, be careful when using the EXEC statement with UNIX commands or scripts that have high overhead. For example, you would not want to rebuild the kernel inside an xglance EXEC statement.

EXEC Examples

In the following example, EXEC executes the UNIX `mailx` command at every interval.

```
EXEC "echo 'xglance mailed you a message' | mailx root"
```

In the following example, EXEC executes the UNIX `mailx` command only when the `gbl_disk_util_peak` metric exceeds 20.

```
IF gbl_disk_util_peak 20 THEN
  EXEC "echo 'xglance detects high disk utilization' | mailx root"
```

GPM Statement

Use the GPM command to have selected GlancePlus windows display whenever conditions that you specify are met.

Syntax:

```
GPM -rpt reportlist
```

The *reportlist* contains the GlancePlus window names for the windows you want to display. In *reportlist*, the window names should be separated by commas. Refer to the Windows List for GlancePlus windows.

GPM Example

```
IF gbl_run_queue 3 THEN
  GPM rpt CpuGraph
```

IF Statement

Use the IF statement to test *conditions* you define in the Adviser syntax.

Syntax:

```
IF condition THEN statement [ELSE statement]
```

The IF statement tests the *condition*. If true, the *statement* after the THEN is executed. If the *condition* is false, then the action depends on the optional ELSE clause.

If an ELSE clause has been specified, the *statement* following it is executed. Otherwise, the IF statement does nothing. The *statement* can be a COMPOUND Statement which tells the Adviser to execute multiple statements.

IF Example

```
IF gbl_cpu_total_util 90 THEN
    PRINT "The CPU is running hot at: ", gbl_cpu_total_util
ELSE IF gbl_cpu_total_util < 20 THEN
    PRINT "The CPU is idling at: ", gbl_cpu_total_util
```

In this example, the IF statement is checking the condition (gbl_cpu_total_util 90). If the condition is true, then "The CPU is running hot at: " is displayed on `stdout` along with the % of CPU used.

If the (gbl_cpu_total_util 90) condition is false, ELSE IF goes to the next line and checks the condition (gbl_cpu_total_util < 20). If that condition is true, then "The CPU is idling at: " is displayed on `stdout` along with the % of CPU used.

LOOP Statement

Use LOOP statements to find information about your system. For example, you can find the process that uses the highest percentage of CPU or the swap area that is being utilized most. You find this information with the LOOP statement and with corresponding statements that use metric names for the system conditions on which you are gathering information. Check which LOOP statements are available on your system.

Syntax:

```
{APPLICATION, APP, CPU, DISK, FILESYSTEM, FS, LAN, LOGICALVOLUME, LV, NETIF, NFS,
NFS_OP, PROCESS, PROC, PROC_FILE, PROC_REGION, SWAP, TRANSACTION, TT, TTBIN,
TT_CLIENT, TT_INSTANCE, TT_UDM, TT_INSTANCE_CLIENT, TT_INSTANCE_UDM, TT_CLIENT_UDM }
LOOP statement
```

A LOOP can be nested within other syntax statements, but you can only nest up to five levels. The *statement* may be a COMPOUND statement which contains a block of statements to be executed on each iteration of the loop. A BREAK statement allows the escape from a LOOP statement.

If you have a LOOP statement in your syntax for collecting specific data and there is no corresponding metric data on your system, the Adviser skips that LOOP and continues to the next syntax statement or instruction. For example, if you have defined a LOGICAL VOLUME LOOP, but have no logical volumes on your system, the Adviser skips that LOGICAL VOLUME LOOP and continues to the next syntax statement.

Loops that do not exist on your platform will generate a syntax error.

As LOOP statements iterate through each interval, the values for the metric used in the statement change. For instance, the following LOOP statement executes the PRINT Statement once for each active application on the system, causing the name of each application to be printed.

```
APP LOOP
    PRINT app_name
```

On a threaded operating system such as HP_UX 11.0, the Adviser supports a THREAD LOOP. A thread loop can be nested inside a process loop in order to examine each thread for a particular process, but the PROC_ (process) metrics should not be referenced directly inside the thread loop. If you would like to use a PROC_ metric inside a thread loop, you should first save it as a local variable in outside process loop. Inside the thread loop, use the local variable. If you do reference a PROC_ metric inside a thread loop, it will return unexpected results (thread information).

An example of this is as follows:

An example of this is:

```
process_to_examine = "glance"
process loop {
    if PROC_PROC_NAME == process_to_examine then {
        proccputime = PROC_CPU_TOTAL_UTIL
        thread loop {
            print "thread ID:",THREAD_THREAD_ID," Thread/proc name: ",
                THREAD_PROC_NAME," Process total CPU=", proccputime
        }
    }
}
```

A thread loop can also exist outside a process loop. In this case, it will examine all threads active on the system. You should not nest a process loop within a thread loop.

NOTE: THREAD metrics are not supported on Sun Solaris systems.

Because LOOP statements are initiated at each interval, use them with discretion due to possible performance implications. This caution is especially appropriate with regards to using nested LOOP statements.

LOOP Statement Examples

To see examples and learn about the different LOOP statements available, select from the list:

[APPLICATION LOOP Example](#)

[CPU LOOP Example](#)

[DISK LOOP Example](#)

[FILE SYSTEM LOOP example](#)

[NETWORK INTERFACE LOOP Example](#)

[NFS BY OPERATION LOOP Example](#)

[PROCESS LOOP Example](#)

[SWAP LOOP Example](#)

[TTBIN LOOP Example](#)

[TT LOOP Example](#)

[TT LOOP ARM Example](#)

APPLICATION LOOP Example

Use the APPLICATION LOOP statement to cycle through all active applications.

You can use global (gbl), table(tbl), or application (app) metrics with the APPLICATION LOOP.

The following example uses an Application LOOP to find the application with the highest CPU for an interval.

```
big_app = ""
highest_cpu = 0
APPLICATION LOOP
  IF (app_cpu_total_util highest_cpu) THEN
    {
      highest_cpu = app_cpu_total_util
      big_app = app_name
    }
  IF (highest_cpu 20) THEN
    YELLOW ALERT "The application ", big_app, " is the highest CPU user at",
      highest_cpu, "%"
```

After finding the application, the Adviser writes a message to the Alarm History window with the app_name and CPU value, if the CPU value is greater than 20.

CPU LOOP Example

Use the CPU LOOP statement to cycle through data about CPU use on your system. You can use global (gbl), table(tbl), or by CPU metrics with the CPU LOOP.

The following example prints the CPU usage percentage for each CPU on your system.

```
Print "-----", glb_stattime, "-----"
CPU LOOP
PRINT "CPU # ", bycpu_id, " used ", bycpu_cpu_total_util, " % CPU"
```

On a system with two CPUs, the resulting output printed for two intervals is:

```
-----10:52:01-----
CPU #          0 used 0.6 % CPU
CPU #          1 used 3.4 % CPU
-----10:52:11-----
CPU #          0 used 0.4 % CPU
CPU #          1 used 2.3 % CPU
```

DISK LOOP Example

Use the DISK LOOP statement to loop through your configured disk devices. When you use this LOOP, the Adviser checks for specific disk information that appears in the IO by Disk window. You can use global (gbl), table(tbl) or by disk metrics with the DISK LOOP.

The following example prints the physical write rate for each disk on your system.

```
PRINT "-----", gbl_stattime, "-----"
DISK LOOP
    PRINT bydisk_devname, " write rate: ", bydisk_phys_write_rate
```

On a system with three disks, the resulting output printed for two intervals is:

```
-----11:00:23-----
/dev/hdisk0      write rate:    2.4
/dev/hdisk1      write rate:    0.0
/dev/cd0         write rate:    0.0
-----11:00:33-----
/dev/hdisk0      write rate:    0.0
/dev/hdisk1      write rate:    0.0
/dev/cd0         write rate:    0.0
```

FILE SYSTEM LOOP Example

The FILE SYSTEM LOOP is designed to loop through configured file systems and allow the Adviser to report on information accessible in the IO By File System Window. You can use global (gbl) , table (tbl) , or IO by file system (fs) metrics with the FILE SYSTEM LOOP.

The following example reports the space utilized for each file system device on a system with three devices.

```
PRINT "-----", gbl_stattime, "-----"
FS LOOP
    PRINT fs_devname, " is ", fs_space_util, "% full at ",fs_max_size," megabytes"
```

The resulting output for two intervals on a system with three file systems is:

```
-----11:11:28-----
/dev/hd4        is  77.9% full at    32 megabytes
/dev/hd2        is  94.9% full at   928 megabytes
/dev/hd9var     is  93.9% full at    56 megabytes
-----11:11:38-----
/dev/hd4        is  77.9% full at    32 megabytes
/dev/hd2        is  94.9% full at   928 megabytes
/dev/hd9var     is  93.6% full at    56 megabytes
```

NFS BY OPERATION LOOP Example

Use the NFS BY OPERATION LOOP to loop through NFS operations performed. When you use this LOOP, the Adviser checks for specific NFS operations that appear in the NFS By Operation window. You can use either global (gbl) , table (tbl) , or by operation metrics with the NFS_OP LOOP.

The following example prints the server and client operations performed:

```
PRINT "-----", gbl_stattime, "-----"
NFS_OP LOOP
    PRINT byop_server_count," server and ",byop_client_count," client ",byop_name,"
    operations performed"
```

On a system performing no activity as an NFS server but with users doing directory listing on another NFS server, the resulting output is:

```
-----14:55:41-----
    0 server and    0 client null          operations performed
    0 server and    2 client getattr       operations performed
    0 server and    0 client setattr       operations performed
    0 server and    0 client root          operations performed
    0 server and   886 client lookup      operations performed
    0 server and   884 client readlink    operations performed
    0 server and    0 client read          operations performed
    0 server and    0 client writecache     operations performed
    0 server and    0 client write         operations performed
    0 server and    0 client create         operations performed
    0 server and    0 client remove        operations performed
```



```

0 server and      0 client rename      operations performed
0 server and      0 client link         operations performed
0 server and      0 client symlink      operations performed
0 server and      0 client mkdir         operations performed
0 server and      0 client rmdir        operations performed
0 server and      28 client readdir    operations performed
0 server and      1 client statfs       operations performed

```

NETWORK INTERFACE LOOP Example

Use the NETWORK INTERFACE LOOP to loop through configured LAN devices and to report on information from the Network by Interface window. You can use global (gbl), table (tbl), or by network interface (bynetif) metrics with the LAN LOOP.

The following example reports on packets for specific LAN names.

```

PRINT "-----", gbl_stattime, "-----"
NETIF LOOP
PRINT bynetif_name, " had ", bynetif_collision, " collisions and ", bynetif_error,
" errors"

```

The resulting output for two intervals on a system with two interfaces is:

```

-----11:40:00-----
lo0      had      0 collisions and      0 errors
en0      had      0 collisions and      0 errors

-----11:40:10-----
lo0      had      0 collisions and      0 errors
en0      had      0 collisions and      0 errors

```

PROCESS LOOP Example

Use the PROCESS LOOP statement to cycle through all active processes.

You can use either global (gbl), table (tbl), or process (proc) metrics with the PROCESS LOOP.

The following example uses a PROCESS LOOP to find the process with the highest CPU for an interval.

```

big_proc_id = 0
big_proc_name = ""
big_proc_cpu = 0
PROCESS LOOP
  IF proc_cpu_total_util big_proc_cpu THEN
    {
      big_proc_cpu = proc_cpu_total_util
      big_proc_name = proc_proc_name
      big_proc_id = proc_proc_id
    }
  IF big_proc_cpu 10 THEN
  YELLOW ALERT "Possible loop, process ", big_proc_name, " pid ", big_proc_id|6|0,
    " using ", big_proc_cpu, " % CPU"

```

SWAP LOOP Example

Use the SWAP LOOP to LOOP through the configured swap areas and allow the Adviser to report on information from the Swap Space Window. You can use table (tbl) or global (gbl) or by swap (byswp) metrics with the SWAP LOOP.

The following example reports on the swap space available on a system with two swap devices.

```

PRINT "-----", gbl_stattime, "-----"
SWAP LOOP
PRINT BYSWP_SWAP_SPACE_NAME, " has ", BYSWP_SWAP_SPACE_USED, " used out of",
  BYSWP_SWAP_SPACE_AVAIL, " megabytes"

```

On a system with one swap area, the output printed for two intervals is:

```

-----15:31:59-----
/dev/hd6      has      37 used out of      128 megabytes

```

```
-----15:32:09-----
/dev/hd6          has      37 used out of  128 megabytes
```

TT LOOP Example

Use the TT LOOP to loop through transaction information that has been recorded during the last interval. When you use this LOOP, the Adviser checks for specific transaction information that appears in the Transaction Tracking window. You can use global (GBL), table (TBL), or transaction tracking (TT) metrics with TT LOOP.

The following example prints the number of completed transactions and the average response time for each registered transaction name on your system.

```
PRINT "-----", gbl_stattime, "-----"
TT LOOP
PRINT tt_name, " had ", tt_count, " transactions with response time "
, tt_wall_time_per_tran, " secs"
```

On a system with four transactions, the resulting output for two intervals is:

```
-----13:24:44-----
First_Transaction  had      1 transactions with response time  1.000355 secs
Second_Transaction had      1 transactions with response time  2.000221 secs
Third_Transaction  had      1 transactions with response time  3.000231 secs
Fourth_Transaction had      0 transactions with response time  0.000000 secs
-----13:24:54-----
First_Transaction  had      3 transactions with response time  1.000383 secs
Second_Transaction had      1 transactions with response time  2.000216 secs
Third_Transaction  had      0 transactions with response time  0.000000 secs
Fourth_Transaction had      0 transactions with response time  0.000000 secs
```

TTBIN LOOP Example

Use the TTBIN LOOP, which must be nested within a TT loop, to loop through the response time bins of each active transaction on your system. When you use this LOOP, the Adviser checks for specific transaction information that appears in the Transaction Graph window. You can use global (gbl), table (tbl), transaction tracking, or transaction tracking bin metrics with the TTBIN LOOP.

The following example prints the response time bins for each transaction name which had any completed transactions during the interval.

```
PRINT "-----", gbl_stattime, "-----"
TT LOOP
IF (tt_count 0) THEN
{
  print "Transaction ", tt_name, " had ", tt_count, " transactions"
  lower_bin_limit = 0
  TTBIN LOOP
  {
    IF (ttbin_trans_count 0) THEN
    print " ", ttbin_trans_count, " were between ", lower_bin_limit, " and ",
    ttbin_upper_range, " seconds"
    lower_bin_limit = ttbin_upper_range
  }
}
}
```

On a system with four transactions, the output printed for two intervals is:

```
-----13:46:31-----
Transaction First_Transaction  had      4 transactions
      2 were between      1.00 and  2.000000 seconds
Transaction Second_Transaction had      1 transactions
      1 were between      2.00 and  3.000000 seconds
Transaction Third_Transaction  had      1 transactions
      1 were between      3.00 and  5.000000 seconds
-----13:46:41-----
Transaction First_Transaction  had      3 transactions
      1 were between      1.00 and  2.000000 seconds
Transaction Second_Transaction had      1 transactions
```

```

1 were between      2.00 and    3.000000 seconds
Transaction Fourth_Transaction had    1 transactions
1 were between      3.00 and    5.000000 seconds

```

TT LOOP ARM Example

With ARM 2.0, the TT_CLIENT, TT_INSTANCE and TT_UDM loops can be nested within a TT LOOP. The TT_CLIENT loop lists the correlated transactions, the TT_INSTANCE loop lists up to 2048 transaction instances, and the TT_UDM loop lists user measurements for a given transaction. . You can use global (gbl), table (tbl) or transaction tracking metrics with the TT LOOP.

Within a TT_CLIENT loop a user can nest a TT_CLIENT_UDM loop to display user measurements on a per correlator basis. A TT_INSTANCE_UDM loop, or TT_INSTANCE_CLIENT loop may be nested within the TT_INSTANCE loop to see correlators or user measurements specific to a given instance.

The examples below show how multiple loops can be used to look at user measurements for any given transaction instance.

Example 1: Look for SLO Violations

```

# The following example loops thru all transactions looking for SLO # violations,
then prints the UDM information for all instances:

```

```

print "-----", GBL_STATTIME, "-----"

```

```

tt loop {
  IF tt_slo_count 0 THEN {
    print " "
    print "SLO violation count:", tt_slo_count,
          " for transaction:", tt_name, " user:", tt_uname,
          " app:", tt_app_name, " threshold: ", tt_slo_threshold
    tt_instance loop {
      starttime = gbl_stattime - gbl_interval
      IF tt_instance_stop_time starttime
      THEN {
        # found a completed instance in the transaction, print info:
        print "instance pid:", tt_instance_proc_id,
              " wall time:", tt_instance_wall_time
        tt_instance_udm loop {
          print " ", tt_instance_user_measurement_name|44,
                " value= ", tt_instance_user_measurement_value
        }
      }
    }
  }
}

```

```

# the following is the output for one interval:

```

```

-----17:19:03-----

SLO violation count:    1 for transaction:Client_tra00      user:gracel
app:Client_App10      threshold:    5.000000
instance pid: 12137 wall time: 13.0407

SLO violation count:    1 for transaction:Server_transaction  user:joe
app:Server_Application threshold:    5.000000
instance pid: 12137 wall time: 13.0358

```

```

Metric #1 - Type 1 is a COUNTER32      value=      32
Metric #2 - Type 4 is a GAUGE32        value=      37
Metric #3 - Type 2 is a COUNTER64      value=    19088743
Metric #4 - Type 9 is a STRING8        value=    String 8
Metric #5 - Type 3 is a COUNTER32/DIVISOR32 value=      2.000
Metric #6 - Type 8 is a NUMERICID64    value=    19088434
The last field is always a STRING32    value=      0
instance pid: 12137 wall time:  3.0291
Metric #1 - Type 1 is a COUNTER32      value=      32
Metric #2 - Type 4 is a GAUGE32        value=      37
Metric #3 - Type 2 is a COUNTER64      value=    19088743
Metric #4 - Type 9 is a STRING8        value=    String 8
Metric #5 - Type 3 is a COUNTER32/DIVISOR32 value=     21.333
Metric #6 - Type 8 is a NUMERICID64    value=    19088434
The last field is always a STRING32    value=      0
instance pid: 12137 wall time:  3.0256
Metric #1 - Type 1 is a COUNTER32      value=      32
Metric #2 - Type 4 is a GAUGE32        value=      37
Metric #3 - Type 2 is a COUNTER64      value=    19088743
Metric #4 - Type 9 is a STRING8        value=    String 8
Metric #5 - Type 3 is a COUNTER32/DIVISOR32 value=     21.333
Metric #6 - Type 8 is a NUMERICID64    value=    19088434
The last field is always a STRING32    value=      0
instance pid: 12137 wall time:  2.0201
Metric #1 - Type 1 is a COUNTER32      value=      32
Metric #2 - Type 4 is a GAUGE32        value=      37
Metric #3 - Type 2 is a COUNTER64      value=    19088743
Metric #4 - Type 9 is a STRING8        value=    String 8
Metric #5 - Type 3 is a COUNTER32/DIVISOR32 value=     21.333
Metric #6 - Type 8 is a NUMERICID64    value=    19088434
The last field is always a STRING32    value=      0
instance pid: 12137 wall time:  1.0101
Metric #1 - Type 1 is a COUNTER32      value=      32
Metric #2 - Type 4 is a GAUGE32        value=      37
Metric #3 - Type 2 is a COUNTER64      value=    19088743
Metric #4 - Type 9 is a STRING8        value=    String 8
Metric #5 - Type 3 is a COUNTER32/DIVISOR32 value=     21.333
Metric #6 - Type 8 is a NUMERICID64    value=    19088434
The last field is always a STRING32    value=      0

```

Example 2: ARM 2.0 syntax

```

# The following example prints info for all completed transactions
# during the interval.

print "-----", GBL_STATTIME, "-----"
-----"

header_printed = 0
tt loop {

```

```

tt_instance loop {
  starttime = GBL_STATTIME - GBL_INTERVAL
  IF TT_INSTANCE_STOP_TIME starttime
  THEN {
    IF header_printed == 0 THEN {
      print " "
      print "TranID  StartTime                StopTime                TranName"
      header_printed = 1
    }
    print TT_TRAN_ID|6, " ", TT_INSTANCE_START_TIME, " ",
          TT_INSTANCE_STOP_TIME, " ", TT_NAME|40
  }
}
}
}

```

the following is the output for one interval:

```

-----17:21:24-----
TranID  StartTime                StopTime                TranName
  3 Wed Jun  3 17:21:07 1998 Wed Jun  3 17:21:20 1998 Client_tra00
  7 Wed Jun  3 17:21:07 1998 Wed Jun  3 17:21:20 1998 Server_transaction
  7 Wed Jun  3 17:21:17 1998 Wed Jun  3 17:21:20 1998 Server_transaction
  7 Wed Jun  3 17:21:17 1998 Wed Jun  3 17:21:20 1998 Server_transaction
  7 Wed Jun  3 17:21:18 1998 Wed Jun  3 17:21:20 1998 Server_transaction
  7 Wed Jun  3 17:21:19 1998 Wed Jun  3 17:21:20 1998 Server_transaction

```

PRINT Statement

Use the PRINT statement to print to stdout data you are collecting. You may want to use the PRINT Statement to log metrics or calculated variables.

Syntax:

```
PRINT printlist
```

PRINT Example

```
PRINT "The Application OTHER has a total CPU of ", other:app_cpu_total_util, "%"
```

When executed, this statement prints a message to the window that initiated GlancePlus like the following:

```
The Application OTHER has a total CPU of 89%
```

SYMPTOM Statement

Syntax:

```

SYMPTOM variable [TYPE = {CPU, DISK, MEMORY, NETWORK}]
RULE measurement {>, <, >=, <=, ==, !=} value PROB probability
[RULE measurement {>, <, >=, <=, ==, !=} value PROB probability]
.
.
.

```

The keywords SYMPTOM and RULE are exclusive for the SYMPTOM statement and cannot be used in other syntax statements. The SYMPTOM statement must be a top-level statement and cannot be nested within any other statement.

variable is a variable name which will be the name of this symptom, as well as a graph title in the Symptom History window. Variable names defined in the SYMPTOM statement can be used in other syntax statements, but the variable value should not be changed in those statements.

TYPE defines the type of symptom and connects the SYMPTOM information to the CPU, Disk, Memory, or Network button on the Main GlancePlus window. The symptom type can only be CPU, Disk, Memory, or Network. However, you can define more than one CPU, Disk, Memory, or Network symptom. For example, if you have two TYPE = CPU symptoms, each with their own set of RULEs, then the symptom with the highest probability determines the color of the CPU button label.

RULE is an option of the SYMPTOM statement and cannot be used independently. You can use as many RULE options within the SYMPTOM statement as you need.

The SYMPTOM variable is evaluated according to the RULEs at each interval.

- *measurement* is the name of a variable or metric that is evaluated as part of the RULE
- *value* is a constant, variable, or metric that is compared to the *measurement*
- *probability* is a numeric constant, variable, or metric

The probabilities for each true SYMPTOM RULE are added together to create a SYMPTOM value. The SYMPTOM value then appears in bar graph form in the Symptom History window. The SYMPTOM value also appears in the Symptom Status window and the Symptom Snapshot window alphanumerically, if the SYMPTOM evaluates to yellow or red.

The sum of all probabilities where the condition between measurement and value is true is the probability that the symptom is occurring.

SYMPTOM Example

Syntax:

```
SYMPTOM CPU_Bottleneck TYPE=CPU
RULE gbl_cpu_total_util 50  PROB 25
RULE gbl_cpu_total_util 85  PROB 25
RULE gbl_cpu_total_util 90  PROB 25
RULE gbl_run_queue        3  PROB 50
```

```
SYMPTOM CPU_Level TYPE=CPU
RULE gbl_cpu_sys_mode_util 40  PROB 25
RULE gbl_cpu_sys_mode_util 50  PROB 25
RULE gbl_cpu_sys_mode_util 60  PROB 25
RULE gbl_cpu_sys_mode_util 70  PROB 50
```

Whichever CPU symptom defined above has the highest total probability (PROB), is the symptom that determines the label color of the CPU button on the GlancePlus Main window.

Symptom Example: Global CPU Bottleneck

```
SYMPTOM Symp_Global_Cpu_Bottleneck TYPE=CPU
RULE gbl_cpu_total_util 50  PROB 25
RULE gbl_cpu_total_util 85  PROB 25
RULE gbl_cpu_total_util 90  PROB 25
RULE gbl_run_queue        3  PROB 75
```

The SYMPTOM statement establishes a new variable called Symp_Global_Cpu_Bottleneck. TYPE=CPU links the SYMPTOM to the CPU button on the Main window.

The new variable receives a probability every update interval which is computed by summing a value according to the RULEs below the SYMPTOM statement.

If the computed probability is between 51 and 90, the CPU button letters on the Main window are turned to yellow for that interval.

- If the probability is 91 or more, then the CPU button letters are turned red.
- If the probability is 50 or less, the CPU button letters are turned to their normal color.

For example, if the CPU utilization (gbl_cpu_total_util) for the interval was 93% and the run queue was 2, then the first three rules would all be true so that 25 would be added to the probability three times. Since the fourth rule would not be true, 75 would NOT be added. Thus Symp_global_cpu_bottleneck variable would have a value of 75

(percent) that interval and the Main screen CPU button letters would be turned yellow (because the probability between 51 and 90).

If there were several RULES which pertain to CPU in the Adviser Syntax and any of them were to achieve a sufficient probability, the CPU button letters turn the appropriate color. If a RULE would cause the letters to turn yellow and another RULE would cause them to turn red, the highest probability (turning red) is reflected on the CPU button.

Interval

An interval is the period of time since the last measurement. GlancePlus evaluates the Adviser SYMPTOMS and ALARMS at each interval. The default interval is 15 seconds. To change the default interval, use the Configure Measurement window.

GlancePlus Messages

This section contains information on various messages you may encounter while running GlancePlus.

Adviser Syntax Messages tells about messages GlancePlus may generate when you edit the Adviser syntax.

Troubleshooting tells where to look when GlancePlus is behaving unpredictably.

Installation Messages tells what to do when you have problems with GlancePlus installation.

Start-up Messages tells what to do when you have completed installation, but you are having problems running GlancePlus.

Adviser Syntax Messages

To view the message, click the category below, then the message number.

General Messages (AS-101 through AS-131)

Alarm Messages (AS-201 through AS-212)

Symptom Messages (AS-301 through AS-306)

Statement Messages (AS-401 through AS-410)

Action Messages (AS-501 through AS-504)

Loop Messages (AS-601 through AS-636)

General Messages (AS-101 through AS-131)

Message AS-101

Message AS-102

Message AS-103

Message AS-104

Message AS-105

Message AS-106

Message AS-107

Message AS-108

Message AS-110

Message AS-111

Message AS-112

Message AS-113

Message AS-114

Message AS-115

Message AS-116

Message AS-117

Message AS-118

Message AS-119

Message AS-120

Message AS-123

Message AS-124

Message AS-125

Message AS-126

Message AS-127

Message AS-129

Message AS-130

Message AS-131

Message AS-101

Syntax checked OK.

The syntax has been checked and has no errors. You can now install the syntax.

Message AS-102

Syntax Error.

The parser has found a character or item it doesn't recognize, such as '@'.

Message AS-103

Expression syntax is not valid.

See Expressions in the help topic "The Adviser" for information on using expressions.

Message AS-104

Metric or variable name is undefined.

Check the metric name in the Help section 'Performance Metric Definitions', or verify that you have already declared the variable used in the statement.

Message AS-105

Cannot assign to a predefined metric name.

You cannot change a metric value with an assignment statement.

Message AS-106

Application or data feed name **variable name is undefined.**

Check to see that you have correctly spelled the application name.

Message AS-107

Variable **variable name is not defined within application or data feed.**

You have entered an incorrect metric name.

Message AS-108

Variable **variable name is not an application or data feed metric.**

The metric name is valid, but not within the application's context.

Message AS-110

A number is expected.

Enter a number, or check your statement.

Message AS-111

A second variable name or constant is expected after compare operator.

The parser has found a compare operator, and expects to find a **variable name** or valid constant immediately following the operator.

Message AS-112

Missing the right parenthesis.

Enter a right parenthesis to complete the statement.

Message AS-113

Comparison operator is illegal.

See Conditions in the help topic "The Adviser" for information on the valid operators for conditions.

Message AS-114

A condition is expected after the left parenthesis.

See Conditions in the help topic "The Adviser" for information on using conditions.

Message AS-115

An integer field width must follow the "" formatting character.

See Printlist in the help topic "The Adviser" for information on formatting integers.

Message AS-116

Formatting specification is not valid.

See Printlist in the help topic "The Adviser" for an explanation of formatting.

Message AS-117

A condition is required.

See Conditions in the help topic "The Adviser" for information on using conditions.

Message AS-118

Quote expected to end literal.

See Constants in the help topic "The Adviser" for information on using literals.

Message AS-119

User variable `variable name` is numeric and cannot be assigned a string value.

See Variables in the help topic "The Adviser" for information on using variables.

Message AS-120

Operation is not allowed for string variable.

The operation specified cannot be performed on a string variable. For example, you may have tried to add (+) two string variables.

Message AS-123

Cannot assign numeric value to string variable `variable name`.

You have defined a string variable and are trying to assign a numeric value to it. Either change the variable to a numeric type or change the assignment value to a string.

Message AS-124

Cannot assign string value to numeric variable `variable name`.

You have defined a numeric variable and are trying to assign a string value to it. Either change the variable to a string type or change the assignment value to a numeric.

Message AS-125

`variable name` is not a global metric.

A global metric is expected here.

Message AS-126

Cannot recognize statement.

See Adviser Syntax Structure in the help topic "The Adviser" for information on valid statements.

Message AS-127

A valid statement is expected here.

See Adviser Syntax Structure in the help topic "The Adviser" for information on valid statements.

Message AS-129

Report not valid on this platform.

You have tried to specify a window with the `rpt` option that is not available on this platform. See the `rpt` section of Customizing GlancePlus Start-Up for a list of available windows.

Message AS-130

Cannot compare strings with numbers.

The condition statement you entered is not valid. See Conditions in the help topic "The Adviser" for more information.

Message AS-131

This LOOP option is not supported on this platform.

See LOOP Statement in the help topic "The Adviser" for information on appropriate LOOP syntax for this platform.

Alarm Messages (AS-201 through AS-212)

Message AS-201

Message AS-202

Message AS-203

Message AS-204

Message AS-205

Message AS-206

Message AS-207

Message AS-208

Message AS-209

Message AS-210

Message AS-211

Message AS-212

Message AS-201

An alarm condition is required.

See ALARM Statement in the help topic "The Adviser" for information on creating alarm statements.

Message AS-202

Alarm options or alarm actions must follow.

You did not specify an action (START, REPEAT, or END) after the last condition clause.

Message AS-203

At least one alarm action is required.

A valid ALARM and condition have been found, but no actions are specified. Add the action to the ALARM statement.

See ALARM Statement in the help topic "The Adviser" for more information.

Message AS-204

Nested ALARM statements are not allowed.

See ALARM Statement in the help topic "The Adviser" for information on creating alarm statements.

Message AS-205

Conditional ALARM statements are not allowed.

You have tried to put an ALARM statement as part of a condition. See ALARM Statement in the help topic "The Adviser" for information on creating ALARM statements.

Message AS-206

Only one START clause is allowed for an ALARM statement.

See ALARM Statement in the help topic "The Adviser" for information on creating ALARM statements.

Message AS-207

Only one REPEAT clause is allowed for an ALARM statement.

See ALARM Statement in the help topic "The Adviser" for information on creating ALARM statements.

Message AS-208

Only one END clause is allowed for an ALARM statement.

See ALARM Statement in the help topic "The Adviser" for information on creating ALARM statements.

Message AS-209

START clause is out of order.

See ALARM Statement in the help topic "The Adviser" for information on correct ALARM syntax.

Message AS-210

REPEAT clause is out of order.

See ALARM Statement in the help topic "The Adviser" for information on correct ALARM syntax.

Message AS-211

END clause is out of order.

See ALARM Statement in the help topic "The Adviser" for information on correct ALARM syntax.

Message AS-212

A specification of time must follow EVERY keyword.

See ALARM Statement in the help topic "The Adviser" for information on correct ALARM syntax.

Symptom Messages (AS-301 through AS-306)

Message AS-301

Message AS-302

Message AS-303

Message AS-304

Message AS-305

Message AS-306

Message AS-301

Symptom type is not valid.

See SYMPTOM Statement in the help topic "The Adviser" for information on creating symptom statements. The TYPE must be CPU, DISK, MEMORY, or NETWORK.

Message AS-302

Duplicate symptom variable variable name.

You must specify a unique **variable name** for the symptom. See SYMPTOM Statement in the help topic "The Adviser" for more information.

Message AS-303

A user variable name is required for SYMPTOM value.

You have specified a metric name or a **variable name** that has already been defined.

Message AS-304

At least one RULE specification is required.

See SYMPTOM Statement in the help topic "The Adviser" for information on creating symptom statements.

Message AS-305

A PROBABILITY keyword is required.

See SYMPTOM Statement in the help topic "The Adviser" for information on creating symptom statements.

Message AS-306

Expected numeric probability value.

See SYMPTOM Statement in the help topic "The Adviser" for information on PROBABILITY.

Statement Messages (AS-401 through AS-410)

Message AS-401

Message AS-402

Message AS-403

Message AS-404

Message AS-405

Message AS-406

Message AS-407

Message AS-409

Message AS-410

Message AS-401

IF stack underflow - cannot interpret statement.

This is an internal error. Save the appropriate information and contact your support representative.

Message AS-402

An assignment must have an equal sign following the variable name.

This error is often caused by a misspelled statement, variable, or metric name. The system thinks that the statement is an assignment statement rather than an intended statement.

It can also be caused by a missing comma in a printlist, and the cursor may appear one line below the line on which the error clause is specified.

Message AS-403

Assignment value is not valid.

The right side of the assignment is invalid.

Message AS-404

ALIAS statement requires a variable name.

See ALIAS Statement in the help topic "The Adviser" for more information on creating ALIAS statements.

Message AS-405

An ALIAS must have an equal sign following the variable name.

See ALIAS Statement in the help topic "The Adviser" for more information on creating ALIAS statements.

Message AS-406

A LITERAL string value is expected for ALIAS value.

See ALIAS Statement in the help topic "The Adviser" for more information on creating ALIAS statements.

Message AS-407

IF statement requires a THEN.

See IF Statement in the help topic "The Adviser" for more information on creating IF statements.

Message AS-409

Duplicate ALIAS declaration for variable name.

You have specified a duplicate user **variable name**. Use a unique name.

Message AS-410

A condition is required for the IF statement.

See IF Statement in the help topic "The Adviser" for more information on creating IF statements.

Action Messages (AS-501 through AS-504)

Message AS-501

Message AS-502

Message AS-504

Message AS-501

A list of print specifications is expected here.

A list of items to display is expected. See PRINT Statement or ALERT Statement in the help topic "The Adviser" or for more information.

Message AS-502

A print specification is expected next.

A list of items to display is expected. See PRINT Statement or ALERT Statement in the help topic "The Adviser" or for more information.

Message AS-504

A print specification is expected here.

A list of items to display is expected. See See PRINT Statement or ALERT Statement in the help topic "The Adviser" or for more information.

Loop Messages (AS-601 through AS-636)

Message AS-601

Message AS-602
Message AS-603
Message AS-604
Message AS-605
Message AS-606
Message AS-607
Message AS-608
Message AS-609
Message AS-610
Message AS-611
Message AS-612
Message AS-613
Message AS-614
Message AS-615
Message AS-616
Message AS-617
Message AS-618
Message AS-619
Message AS-620
Message AS-621
Message AS-622
Message AS-623
Message AS-624
Message AS-625
Message AS-626
Message AS-627
Message AS-628
Message AS-629
Message AS-630
Message AS-633
Message AS-634
Message AS-635
Message AS-636

Message AS-601

A LOOP keyword is expected following the class type.

See LOOP Statement in the help topic "The Adviser" for information on LOOP syntax.

Message AS-602

LOOP statements nested too deep.

You can only nest statements up to five levels. See LOOP Statement in the help topic "The Adviser" for information on LOOP syntax.

Message AS-603

metric name is not an application or a global metric.

You can only use application or global metrics in an APPLICATION LOOP Statement.

Message AS-604

metric name is not a process or a global metric.

You can only use process or global metrics in a PROCESS LOOP Statement.

Message AS-605

metric name is not a by disk or a global metric.

You can only use by disk or global metrics in a DISK LOOP Statement.

Message AS-606

metric name is not a by CPU or a global metric.

You can only use by CPU or global metrics in a CPU LOOP Statement.

Message AS-607

metric name is not a by LAN or a global metric.

You can only use LAN or global metrics in a LAN LOOP Statement.

Message AS-608

metric name is not a by swap or a global metric.

You can only use by swap or global metrics in a SWAP LOOP Statement.

Message AS-609

metric name is not a by file system or a global metric.

You can only use file system or global metrics in a FILE SYSTEM LOOP Statement.

Message AS-610

metric name is not a by logical volume or a global metric.

You can only use logical volume or global metrics in a LOGICAL VOLUME LOOP Statement.

Message AS-611

metric name is not a by system calls or a global metric.

You can only use system call or global metrics in a SYSTEM CALL LOOP Statement.

Message AS-612

metric name is not a PRM or a global metric.

You can only use PRM or global metrics in a PRM LOOP Statement.

Message AS-613

metric name is not a by TT or a Global metric.

You can only use TT or global metrics in a TT LOOP Statement

Message AS-614

metric name is not a TT Bin, TT, or a Global metric.

You can only use TT BIN, TT or global metrics in a TTBIN LOOP Statement.

Message AS-615

TT Bin loop must be in a TT loop.

A TT Bin loop can only be nested in a TT LOOP Statement.

Message AS-616

metric name is not a Process File, Process, or Global metric.

You can only use process or global metrics in a PROCESS LOOP Statement.

Message AS-617

Process File loop must be inside a Process loop.

A Process File loop can only be nested in a PROCESS LOOP Statement.

Message AS-618

metric name is not a Process Memory Region, Process, or a Global metric.

You can only use Process Memory Region, Process or global metrics in a PROC_REGION LOOP Statement.

Message AS-619

A Process Memory Region loop must be inside a Process loop.

A Process Memory Region loop can only be nested in a PROCESS LOOP Statement.

Message AS-620

metric name is not a by NFS, or a Global metric.

You can only use by NFS or global metrics in a FILE SYSTEM LOOP Statement.

Message AS-621

metric name is not a NFS by System Operation, NFS, or a Global metric.

You can only use NFS by System Operation, NFS or global metrics in a NFS_BYSYS_OP LOOP Statement.

Message AS-622

NFS by System Operations loop must be inside a NFS loop.

NFS by System Operations loop can only be nested in a NFS LOOP Statement.

Message AS-623

metric name is not a by NFS Operation or a Global metric.

You can only use by NFS Operation or global metrics in a NFS_OP LOOP Statement.

Message AS-624

metric name is not a by CNODE or a Global metric.

You can only use by CNODE or global metrics in a CNODE LOOP Statement.

Message AS-625

Disk Detail loop must be inside a Disk loop.

Disk Detail loop can only be nested inside a DISK LOOP Statement.

Message AS-626

metric name is not a Disk Detail, by Disk, or a Global metric.

You can only use Disk Detail, by Disk or global metrics in a DISK_DETAIL LOOP Statement.

Message AS-627

File System Detail loop must be inside a File System loop.

File System Detail loop can only be nested in a FILE SYSTEM LOOP Statement.

Message AS-628

metric name is not a File System Detail, by File System, or Global metric.

You can only use File System Detail, by File System or global metrics in a FS_DETAIL LOOP Statement.

Message AS-629

Logical Volume Detail loop must be inside a Logical Volume loop.

Logical Volume Detail loop can only be nested in a LOGICAL VOLUME LOOP Statement.

Message AS-630

metric name is not a Logical Volume Detail, by Logical Volume, or a Global metric.

You can only use Logical Volume Detail loop, by Logical Volume or global metrics in a LV_DETAIL LOOP Statement.

Message AS-633

metric name is not a Process System Call, Process, or a Global metric.

You can only use Process System Call, Process or global metrics in a PROC_SYSCALL LOOP Statement.

Message AS-634

Process System Call loop must be inside a Process loop.

Process System Call loop can only be nested in a PROCESS LOOP Statement.

Message AS-635

A BREAK keyword must be inside a LOOP statement.

See LOOP Statement for information on proper LOOP statement syntax.

Message AS-636

metric name is not a by LANIF, or a Global metric.

You can only use LANIF or global metrics in a LAN LOOP Statement.

Troubleshooting

The `xglance` program writes its error and status messages to `stderr`, which by default is the terminal window that you ran `xglance` from. Error and status information may also be written to a file named:

```
/var/opt/perf/status.mi
```

If GlancePlus is behaving unexpectedly, this information can assist in troubleshooting problems.

Installation Messages

This section lists messages you may encounter when installing GlancePlus.

ERROR: This script must be run by superuser (root).
Your user id is: *username*

ERROR: Problem copying [**app-defaults,parm,gkey**] file to **/var/opt/perf**
Check that the files are in **/opt/perf/newconfig/*** and that they are readable for copying. Also, check that **/var/opt/perf** is writable and has sufficient filespace.

ERROR: The **midaemon** process must be shut down prior to installation of the **MI** fileset. Use the command **/opt/perf/bin/midaemon -T** to terminate performance tool processes and then reattempt **swinstall** of this fileset.
The **-T** option should be used only before an **MI** fileset installation. Refer to the **midaemon** man pages for details.

Start-up Messages

This section lists messages you may encounter when starting GlancePlus.

Running in Background Mode

During Initialization Start-up

Running in Background Mode

If you attempt to run GlancePlus in background mode from the command line, you may see a message similar to the following

```
$ xglance &&  
[1] 15080  
$  
[1] + Stopped (tty output)  
$
```

This message indicates that the **tty** line discipline is set to **tostop**. This means that any background process attempting to write to the associated standard output will be blocked. GlancePlus writes initialization messages to standard output. You can verify the **tty** line discipline options by typing:

```
$ stty -a
```

Refer to **man** pages for details on **stty**.

Once you have determined the state of the line discipline, you have at least two options available:

1. You can unset the **tty** blocking option and execute your GlancePlus background mode command by typing:

```
$ stty -tostop  
$ xglance &&
```

2. You can keep your **tty** blocking option and redirect the standard output of GlancePlus to **/dev/null** by typing:

```
$ xglance 1>/dev/null &&
```

If GlancePlus is blocked on **stderr** because of the **tty** line discipline **tostop** blocking option, you can resume the blocked `xglance` by moving the process to the foreground by typing:

\$ fg

Refer to **man** pages for shell-specific details.

During Initialization Start-up

As GlancePlus starts up, its progress through various phases of initialization is listed on your screen. Paying attention to the phase in which warning or error messages occur is key to solving a problem that may occur during startup.

Following is an example of the messages that print to your workstation or terminal window on a successful GlancePlus startup:

```
Connecting to Display...  
Connected to X Display  
Enabling Measurement...  
Measurement Enabled  
Loading Configuration (30)...  
Configuration Loaded  
Parse Adviser Syntax...  
Syntax Parsed  
Initialize Shells...  
Shells Initialized
```

Typically, initialization warning or error messages occur in the step they are related to. For instance, if you have not specified the export of your DISPLAY correctly, GlancePlus fails while **Connecting to Display...** and you will see an abort message.

Messages Before Connecting to Display

You may see warnings during parsing of the parm file, which contains your application definitions. A message like this is typically only a warning and should not keep GlancePlus from starting up. You can find information about proper syntax for application definitions in online help in the Users Guide under the topic "Defining Applications."

Messages While Connecting to Display

You may encounter an initialization message such as

```
Connecting to Display...  
Error: Can't open display: workstationid:0.0
```

This error is caused by the X software telling you it can't find the display with the name **workstationid:0.0**. Possible problems could be:

- • You have miskeyed the workstation name.
- • You are using the export method for a different shell than the one you are running. Following are different shell commands for connecting to a display.

```
posix shell: export DISPLAY=workstationid:0.0  
c-shell: setenv DISPLAY workstationid:0.0
```

- • You need to qualify the name with a domain and organization (or even use the IP address of the workstation instead of the name - just in case that system does not have your workstation or X-terminal configured in its **/etc/hosts** file).

If you still can't open your display after checking the steps above, try starting GlancePlus with the **-display** parameter:

```
xglance -display workstationid:0.0
```

- • You may also see an initialization such as

```
Connecting to Display...  
Xlib: connection to workstationid refused by server  
Xlib: Client is not authorized to connect to Server  
Error: Can't open display: workstationid
```

The error above is caused by your workstation or X-terminal software telling you that the system on which you are trying to run GlancePlus is not allowed to export to your display.

To fix this error, enter **xhost +**, which allows any host to export GlancePlus to your display. For more information about exporting displays, read the man page for **xhost**.

- • You may also see an initialization such as

Connecting to Display..

Warning: Cannot convert string "<Key>InsertChar" to type Virtual Binding

Warning: Cannot convert string "<Key>DeleteChar" to type Virtual Binding

Warning: Cannot convert string "<Key>InsertLine" to type Virtual Binding

Warning: Cannot convert string "<Key>DeleteLine" to type Virtual Binding

To fix this problem, make sure that the file **/usr/lib/X11/XKeysymDB** includes the following lines. The lines are typically located at the end of the file.

```
hpInsertLine:1000FF70
```

```
hpDeleteLine:1000FF71
```

```
hpInsertChar:1000FF72
```

```
hpDeleteChar:1000FF73
```

- • Another initialization message you may see is

Connecting to Display...

Warning: translation table syntax error: Unknown keysym name:

```
    osfActivate:ManagerParentActivate()'
```

.

.

.

Connected to Display

This message is caused by the system being unable to either find or read the following file, which contains a keystroke translation table:

```
/usr/lib/X11/XKeysymDB
```

Check for existence of the file and proper permissions. This file is normally on every system.

Messages After Connecting to Display

- • You might see an error message pertaining to the **midaemon** after the following line:

Enabling Measurement...

If GlancePlus has trouble initiating the **midaemon**, you will see an error message that starts with "**mi**" displayed to the **stdout** and placed in the **/var/opt/perf/status.mi** file. The **midaemon** error messages and work-arounds are documented in the man page of **midaemon**.

If you encounter a **midaemon** error, logon as **root**, then try to run the **midaemon** on its own by entering **midaemon** at the command line. Once the **midaemon** is running, try to run GlancePlus again.

To start the **midaemon** so that exiting GlancePlus does not terminate it, enter:

```
/opt/perf/bin/midaemon -p .
```

To make sure the **midaemon** is running, enter **perfstat**.

Look for the following line in the resulting display to verify that **midaemon** is running:

```
/opt/perf/bin/midaemon
```

If it is running, enter **xglance** and see if GlancePlus starts successfully.

- • You may see the following message after **Syntax Parsed...**

```
***** FATAL ERROR *****
```

```
Module: wm.c Line: 141
```

**Message: Could not load fontname font.
 Use the resource Gpmfontname to select one.
 ***** FATAL ERROR *******

This happens when one of the fonts used by GlancePlus is not available on your system. Just add the following line to your .Xdefaults file:

Gpmfontname:font description

fontname is the resource shown in the error message *font description* is the full font description, such as "-adobe-helvetica-medium-r-normal--17-120-100-100-p-88-iso8859-1".

See the */var/opt/perf/Gpm* file for an example of each font's format.

- You may see the following messages as GlancePlus attempts to **Initialize Shells...**

Initialize Shells...

Could not load private color cells - trying shared cells

GlancePlus tries to obtain its own private colors so that you can modify them within GlancePlus. However, if that doesn't work, it obtains colors that can be shared with other applications.

You can also use the *xglance -sharedclr* startup option. This option shares the color scheme with applications that are already running and disables the GlancePlus Configure Color window.

For more information see *-sharedclr* in the man page for *xglance*.

Successful Initialization

Typically, once you see the **Shells Initialized** message no initialization errors occur. If you do see an error after this message, it may be a problem that requires help from your system administrator or your support channel.

Performance Metrics

APP_ACTIVE_APP

The number of applications that had processes active (consuming cpu resources) during the interval.

APP_ACTIVE_PROC

An active process is one that exists and consumes some CPU time. *APP_ACTIVE_PROC* is the sum of the alive-process-time/interval-time ratios of every process belonging to an application that is active (uses any CPU time) during an interval.

The following diagram of a four second interval showing two processes, A and B, for an application should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

	Seconds			
	1	2	3	4
Proc				
A	live	live	live	live
B	live/CPU	live/CPU	live	dead

Process A is alive for the entire four second interval, but consumes no CPU. A's contribution to *APP_ALIVE_PROC* is $4 \times \frac{1}{4}$. A contributes $0 \times \frac{1}{4}$ to *APP_ACTIVE_PROC*. B's contribution to *APP_ALIVE_PROC* is $3 \times \frac{1}{4}$. B contributes $2 \times \frac{1}{4}$ to *APP_ACTIVE_PROC*. Thus, for this interval, *APP_ACTIVE_PROC* equals 0.5 and *APP_ALIVE_PROC* equals 1.75.

Because a process may be alive but not active, *APP_ACTIVE_PROC* will always be less than or equal to *APP_ALIVE_PROC*.

This metric indicates the number of processes in an application group that are competing for the CPU. This metric is useful, along with other metrics, for comparing loads placed on the system by different groups of processes.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_ALIVE_PROC

An alive process is one that exists on the system. APP_ALIVE_PROC is the sum of the alive-process-time/interval-time ratios for every process belonging to a given application.

The following diagram of a four second interval showing two processes, A and B, for an application should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

```
----- Seconds -----
      1         2         3         4
Proc
----
A   live      live      live      live
B   live/CPU  live/CPU  live      dead
```

Process A is alive for the entire four second interval but consumes no CPU. A's contribution to APP_ALIVE_PROC is $4 \times \frac{1}{4}$. A contributes $0 \times \frac{1}{4}$ to APP_ACTIVE_PROC. B's contribution to APP_ALIVE_PROC is $3 \times \frac{1}{4}$. B contributes $2 \times \frac{1}{4}$ to APP_ACTIVE_PROC. Thus, for this interval, APP_ACTIVE_PROC equals 0.5 and APP_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, APP_ACTIVE_PROC will always be less than or equal to APP_ALIVE_PROC.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_COMPLETED_PROC

The number of processes in this group that completed during the interval.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_CPU_SYS_MODE_TIME

The time, in seconds, during the interval that the CPU was in system mode for processes in this group.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HP-UX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_SYS_MODE_UTIL

The percentage of time during the interval that the CPU was used in system mode for processes in this group.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

High system CPU utilizations are normal for IO intensive groups. Abnormally high system CPU utilization can indicate that a hardware problem is causing a high interrupt rate. It can also indicate programs that are not making efficient system calls. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_TOTAL_TIME

The total CPU time, in seconds, devoted to processes in this group during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_TOTAL_UTIL

The percentage of the total CPU time devoted to processes in this group during the interval. This indicates the relative CPU load placed on the system by processes in this group.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

Large values for this metric may indicate that this group is causing a CPU bottleneck. This would be normal in a computation-bound workload, but might mean that processes are using excessive CPU time and perhaps looping.

If the "other" application shows significant amounts of CPU, you may want to consider tuning your parm file so that process activity is accounted for in known applications.

```
APP_CPU_TOTAL_UTIL =  
    APP_CPU_SYS_MODE_UTIL +  
    APP_CPU_USER_MODE_UTIL
```

NOTE: On Windows, the sum of the APP_CPU_TOTAL_UTIL metrics may not equal GBL_CPU_TOTAL_UTIL. Microsoft states that "this is expected behavior" because the GBL_CPU_TOTAL_UTIL metric is taken from the NT performance library Processor objects while the APP_CPU_TOTAL_UTIL metrics are taken from the Process objects. Microsoft states that there can be CPU time accounted for in the Processor system objects that may not be seen in the Process objects. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_TOTAL_UTIL_CUM

The average CPU time per interval for processes in this group over the cumulative collection time, or since the last PRM configuration change on HP-UX.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_USER_MODE_TIME

The time, in seconds, that processes in this group were in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_CPU_USER_MODE_UTIL

The percentage of time that processes in this group were using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

High user mode CPU percentages are normal for computation-intensive groups. Low values of user CPU utilization compared to relatively high values for `APP_CPU_SYS_MODE_UTIL` can indicate a hardware problem or improperly tuned programs in this group.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HPUX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with “-ignore_mt” by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

APP_DISK_PHYS_IO_RATE

The number of physical IOs per second for processes in this group during the interval.

APP_DISK_PHYS_READ

The number of physical reads for processes in this group during the interval.

APP_DISK_PHYS_READ_RATE

The number of physical reads per second for processes in this group during the interval.

APP_DISK_PHYS_WRITE

The number of physical writes for processes in this group during the interval.

APP_DISK_PHYS_WRITE_RATE

The number of physical writes per second for processes in this group during the interval.

APP_INTERVAL

The amount of time in the interval.

APP_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

APP_IO_BYTE

The number of characters (in KB) transferred for processes in this group to all devices during the interval. This includes IO to disk, terminal, tape and printers.

APP_IO_BYTE_RATE

The number of characters (in KB) per second transferred for processes in this group to all devices during the interval. This includes IO to disk, terminal, tape and printers.

APP_MAJOR_FAULT

The number of major page faults that required a disk IO for processes in this group during the interval.

APP_MAJOR_FAULT_RATE

The number of major page faults per second that required a disk IO for processes in this group during the interval.

APP_MEM_RES

On Unix systems, this is the sum of the size (in MB) of resident memory for processes in this group that were alive at the end of the interval. This consists of text, data, stack, and shared memory regions.

On HP-UX, since PROC_MEM_RES typically takes shared region references into account, this approximates the total resident (physical) memory consumed by all processes in this group.

On all other Unix systems, this is the sum of the resident memory region sizes for all processes in this group. When the resident memory size for processes includes shared regions, such as shared memory and library text and data, the shared regions are counted multiple times in this sum. For example, if the application contains four processes that are attached to a 500MB shared memory region that is all resident in physical memory, then 2000MB is contributed towards the sum in this metric. As such, this metric can overestimate the resident memory being used by processes in this group when they share memory regions.

Refer to the help text for PROC_MEM_RES for additional information.

On Windows, this is the sum of the size (in MB) of the working sets for processes in this group during the interval. The working set counts memory pages referenced recently by the threads making up this group. Note that the size of the working set is often larger than the amount of pagefile space consumed.

APP_MEM_UTIL

On Unix systems, this is the approximate percentage of the system's physical memory used as resident memory by processes in this group that were alive at the end of the interval. This metric summarizes process private and shared memory in each application.

On Windows, this is an estimate of the percentage of the system's physical memory allocated for working set memory by processes in this group during the interval.

On HP-UX, this consists of text, data, stack, as well the process' portion of shared memory regions (such as, shared libraries, text segments, and shared data). The sum of the shared region pages is typically divided by the number of references.

On Unix systems, each application's total resident memory is summed. This value is then divided by the summed total of all applications resident memory and then multiplied by the ratio of available user memory versus total physical memory to arrive at a calculated percentage of the total physical memory. It must be remembered, however, that this is a calculated metric that shows the approximate percentage of the physical memory used as resident memory by the processes in this application during the interval.

On Windows, the sum of the working set sizes for each process in this group is kept as APP_MEM_RES. This value is divided by the sum of APP_MEM_RES for all applications defined on the system to come up with a ratio of this application's working set size to the total. This value is then multiplied by the ratio of available user memory versus total physical memory to arrive at a calculated percent of total physical memory.

APP_MEM_VIRT

On Unix systems, this is the sum (in MB) of virtual memory for processes in this group that were alive at the end of the interval. This consists of text, data, stack, and shared memory regions.

On HP-UX, since PROC_MEM_VIRT typically takes shared region references into account, this approximates the total virtual memory consumed by all processes in this group.

On all other Unix systems, this is the sum of the virtual memory region sizes for all processes in this group. When the virtual memory size for processes includes shared regions, such as shared memory and library text and data, the shared regions are counted multiple times in this sum. For example, if the application contains four processes that are attached to a 500MB shared memory region, then 2000MB is reported in this metric. As such, this metric can overestimate the virtual memory being used by processes in this group when they share memory regions.

On Windows, this is the sum (in MB) of paging file space used for all processes in this group during the interval. Groups of processes may have working set sizes (APP_MEM_RES) larger than the size of their pagefile space.

APP_MINOR_FAULT

The number of minor page faults satisfied in memory (a page was reclaimed from one of the free lists) for processes in this group during the interval.

APP_MINOR_FAULT_RATE

The number of minor page faults per second satisfied in memory (pages were reclaimed from one of the free lists) for processes in this group during the interval.

APP_NAME

The name of the application (up to 20 characters). This comes from the parm file where the applications are defined.

The application called "other" captures all processes not aggregated into applications specifically defined in the parm file. In other words, if no applications are defined in the parm file, then all process data would be reflected in the "other" application.

APP_NUM

The sequentially assigned number of this application.

APP_PRI

On Unix systems, this is the average priority of the processes in this group during the interval.

On Windows, this is the average base priority of the processes in this group during the interval.

APP_PROC_RUN_TIME

The average run time for processes in this group that completed during the interval.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

APP_SAMPLE

The number of samples of process data that have been averaged or accumulated during this sample.

APP_TIME

The end time of the measurement interval.

BYCPU_ACTIVE

Indicates whether or not this CPU is online. A CPU that is online is considered active.

For HP-UX and certain versions of Linux, the sar(1M) command allows you to check the status of the system CPUs.

For SUN and DEC, the commands psrinfo(1M) and psradm(1M) allow you to check or change the status of the system CPUs.

For AIX, the pstat(1) command allows you to check the status of the system CPUs.

BYCPU_CPU_CLOCK

The clock speed of the CPU in the current slot. The clock speed is in MHz for the selected CPU.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be "na", some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

On Linux, this value is always rounded up to the next MHz.

BYCPU_CPU_INTERRUPT_TIME

The time, in seconds, that this CPU was performing interrupt processing during the interval. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_TIME_CUM

The time, in seconds, that this CPU was performing interrupt processing over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_UTIL

The percentage of time that this CPU was performing interrupt processing during the interval. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_INTERRUPT_UTIL_CUM

The percentage of time that this CPU was performing interrupt processing over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_TIME

The time, in seconds, that this CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_TIME_CUM

The time, in seconds, that this CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_UTIL

The percentage of time that this CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding

ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_NICE_UTIL_CUM

The average percentage of time that this CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HPUX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_TIME

The time, in seconds, that this CPU was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode. On platforms other than HPUX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_TIME_CUM

The time, in seconds, that this CPU was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HPUX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt"

by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_UTIL

The percentage of time that this CPU was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode. On platforms other than HPUX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the `--ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `--ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_SYS_MODE_UTIL_CUM

The percentage of time that this CPU was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HPUX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HPUX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the `--ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `--ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_TIME

The total time, in seconds, that this CPU was not idle during the interval. On platforms other than HPUX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the `--ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `--ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_TIME_CUM

The total time, in seconds, that this CPU was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_UTIL

The percentage of time that this CPU was not idle during the interval. On platforms other than HP-UX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TOTAL_UTIL_CUM

The average percentage of time that this CPU was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_TYPE

The type of processor in the current slot.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be "na", some will be based on what is visible at startup time. All

information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

BYCPU_CPU_USER_MODE_TIME

The time, in seconds, during the interval that this CPU was in user mode.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_TIME_CUM

The time, in seconds, that this CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_UTIL

The percentage of time that this CPU was in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_CPU_USER_MODE_UTIL_CUM

The average percentage of time that this CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last. On platforms other than HP-UX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

BYCPU_ID

The ID number of this CPU. On some Unix systems, such as SUN, CPUs are not sequentially numbered.

BYCPU_INTERRUPT

The number of device interrupts for this CPU during the interval.

On HP-UX, a value of "na" is displayed on a system with multiple CPUs.

BYCPU_INTERRUPT_RATE

The average number of device interrupts per second for this CPU during the interval.

On HP-UX, a value of "na" is displayed on a system with multiple CPUs.

BYCPU_STATE

A text string indicating the current state of a processor.

On HP-UX, this is either "Enabled", "Disabled" or "Unknown". On AIX, this is either "Idle/Offline" or "Online". On all other systems, this is either "Offline", "Online" or "Unknown".

BYDSK_AVG_REQUEST_QUEUE

The average number of IO requests that were in the wait and service queues for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For example, if 4 intervals have passed with average queue lengths of 0, 2, 0, and 6, then the average number of IO requests over all intervals would be 2.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_AVG_SERVICE_TIME

The average time, in milliseconds, that this disk device spent processing each disk request during the interval. For example, a value of 5.14 would indicate that disk requests during the last interval took on average slightly longer than five one-thousandths of a second to complete for this device.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

This is a measure of the speed of the disk, because slower disk devices typically show a larger average service time. Average service time is also dependent on factors such as the distribution of I/O requests over the interval and their locality. It can also be influenced by disk driver and controller features such as I/O merging and command queueing. Note that this service time is measured from the perspective of the kernel, not the disk device itself. For example, if a disk device can find the requested data in its cache, the average service time could be quicker than the speed of the physical disk hardware.

This metric can be used to help determine which disk devices are taking more time than usual to process requests.

BYDSK_DEVNAME

The name of this disk device.

On HP-UX, the name identifying the specific disk spindle is the hardware path which specifies the address of the hardware components leading to the disk device.

On SUN, these names are the same disk names displayed by "iostat".

On AIX, this is the path name string of this disk device. This is the fsname parameter in the mount(1M) command. If more than one file system is contained on a device (that is, the device is partitioned), this is indicated by an asterisk ("**") at the end of the path name.

On OSF1, this is the path name string of this disk device. This is the file-system parameter in the mount(1M) command.

On Windows, this is the unit number of this disk device.

BYDSK_DEVNO

Major / Minor number of the device.

BYDSK_DIRNAME

The name of the file system directory mounted on this disk device. If more than one file system is mounted on this device, "Multiple FS" is seen.

BYDSK_ID

The ID of the current disk device.

BYDSK_INTERVAL

The amount of time in the interval.

BYDSK_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_PHYS_BYTE

The number of KBs of physical IOs transferred to or from this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_BYTE_RATE

The average KBs per second transferred to or from this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_BYTE_RATE_CUM

The average number of KBs per second of physical reads and writes to or from this disk device over the cumulative collection time.

On Unix systems, this includes all types of physical disk IOs including file system, virtual memory, and raw IOs.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_PHYS_IO

The number of physical IOs for this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw reads.

BYDSK_PHYS_IO_RATE

The average number of physical IO requests per second for this disk device during the interval.

On Unix systems, all types of physical disk IOs are counted, including file system IO, virtual memory and raw IO.

BYDSK_PHYS_IO_RATE_CUM

The average number of physical reads and writes per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_PHYS_READ

The number of physical reads for this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On AIX, this is an estimated value based on the ratio of read bytes to total bytes transferred. The actual number of reads is not tracked by the kernel. This is calculated as

```
BYDSK_PHYS_READ =  
  BYDSK_PHYS_IO *  
  (BYDSK_PHYS_READ_BYTE /  
   BYDSK_PHYS_IO_BYTE)
```

BYDSK_PHYS_READ_BYTE

The KBs transferred from this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_READ_BYTE_RATE

The average KBs per second transferred from this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_READ_BYTE_RATE_CUM

The average number of KBs per second of physical reads from this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_PHYS_READ_RATE

The average number of physical reads per second for this disk device during the interval.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On AIX, this is an estimated value based on the ratio of read bytes to total bytes transferred. The actual number of reads is not tracked by the kernel. This is calculated as

$$\text{BYDSK_PHYS_READ_RATE} = \text{BYDSK_PHYS_IO_RATE} * (\text{BYDSK_PHYS_READ_BYTE} / \text{BYDSK_PHYS_IO_BYTE})$$

BYDSK_PHYS_READ_RATE_CUM

The average number of physical reads per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_PHYS_WRITE

The number of physical writes for this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On AIX, this is an estimated value based on the ratio of write bytes to total bytes transferred because the actual number of writes is not tracked by the kernel. This is calculated as

$$\text{BYDSK_PHYS_WRITE} = \text{BYDSK_PHYS_IO} * (\text{BYDSK_PHYS_WRITE_BYTE} / \text{BYDSK_PHYS_IO_BYTE})$$

BYDSK_PHYS_WRITE_BYTE

The KBs transferred to this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_WRITE_BYTE_RATE

The average KBs per second transferred to this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system, virtual memory, and raw IO.

BYDSK_PHYS_WRITE_BYTE_RATE_CUM

The average number of KBs per second of physical writes to this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_PHYS_WRITE_RATE

The average number of physical writes per second for this disk device during the interval.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On AIX, this is an estimated value based on the ratio of write bytes to total bytes transferred. The actual number of writes is not tracked by the kernel. This is calculated as

```
BYDSK_PHYS_WRITE_RATE =  
    BYDSK_PHYS_IO_RATE *  
    (BYDSK_PHYS_WRITE_BYTE /  
     BYDSK_PHYS_IO_BYTE)
```

BYDSK_PHYS_WRITE_RATE_CUM

The average number of physical writes per second for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYDSK_QUEUE_0_UTIL

The percentage of intervals during which there were no IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 1.5, 0, and 3, then the value for this metric would be 50% since 50% of the intervals had a zero queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_2_UTIL

The percentage of intervals during which there were 1 or 2 IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 1, 0, and 2, then the value for this metric would be 50% since 50% of the intervals had a 1-2 queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_4_UTIL

The percentage of intervals during which there were 3 or 4 IO requests waiting to use this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 3, 0, and 4, then the value for this metric would be 50% since 50% of the intervals had a 3-4 queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_8_UTIL

The percentage of intervals during which there were between 5 and 8 IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 8, 0, and 5, then the value for this metric would be 50% since 50% of the intervals had a 5-8 queue length.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_QUEUE_X_UTIL

The percentage of intervals during which there were more than 8 IO requests pending for this disk device over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For example if 4 intervals have passed (that is, 4 screen updates) and the average queue length for these intervals was 0, 9, 0, and 10, then the value for this metric would be 50% since 50% of the intervals had queue length greater than 8.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_REQUEST_QUEUE

The average number of IO requests that were in the wait queue for this disk device during the interval. These requests are the physical requests (as opposed to logical IO requests).

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

BYDSK_TIME

The time of day of the interval.

BYDSK_UTIL

On HP-UX, this is the percentage of the time during the interval that the disk device had IO in progress from the point of view of the Operating System. In other words, the utilization or percentage of time busy servicing requests for this device.

On the non-HP-UX systems, this is the percentage of the time that this disk device was busy transferring data during the interval.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

This is a measure of the ability of the IO path to meet the transfer demands being placed on it. Slower disk devices may show a higher utilization with lower IO rates than faster disk devices such as disk arrays. A value of greater than 50% utilization over time may indicate that this device or its IO path is a bottleneck, and the access pattern of the workload, database, or files may need reorganizing for better balance of disk IO load.

BYLS_BOOT_TIME

On vMA, for a host and logical system the metric is the date and time when the system was last booted. The value is NA for resource pool. Note that this date is obtained from the VMware API as an already formatted string and may not conform to the expected localization.

BYLS_CLUSTER_NAME

On vMA, for a host and resource pool it is the name of the cluster to which the host belongs to when it is managed by virtual centre. For a logical system, the value is NA.

BYLS_CPU_CLOCK

On vMA, for a host and logical system, it is the clock speed of the CPUs in MHz if all of the processors have the same clock speed. For a resource pool the value is NA.

BYLS_CPU_CYCLE_ENTL_MAX

On vMA, for a host, logical system and resource pool this value indicates the maximum processor capacity, in MHz, configured for the entity. If the maximum processor capacity is not configured for the entity, a value of "-3" will be displayed in PA and "ul"(unlimited) in other clients.

On HPUX, the maximum processor capacity, in MHz, configured for this logical system.

BYLS_CPU_CYCLE_ENTL_MIN

On vMA, for a host, logical system and resource pool this value indicates the minimum processor capacity, in MHz, configured for the entity.

On HPUX, the minimum processor capacity, in MHz, configured for this logical system.

BYLS_CPU_CYCLE_TOTAL_USED

On vMA, for host, resource pool and logical system, it is the total time the physical CPUs were utilized during the interval, represented in cpu cycles.

BYLS_CPU_ENTL_EMIN

On vMA, for host, logical system and resource pool the value is "na".

BYLS_CPU_ENTL_MAX

The maximum CPU units configured for a logical system.

On HP-UX HPVM, this metric indicates the maximum percentage of physical CPU that a virtual CPU of this logical system can get.

On AIX SPLPAR, this metric is equivalent to "Maximum Capacity" field of 'lparstat -i' command.

For WPARs, it is the maximum percentage of CPU that a WPAR can have even if there is no contention for CPU. WPAR shares CPU units of its global environment.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a host, the metric is equivalent to total number of cores on the host. For a resource pool and a logical system, this metrics indicates the maximum CPU units configured for it.

BYLS_CPU_ENTL_MIN

The minimum CPU units configured for this logical system.

On HP-UX HPVM, this metric indicates the minimum percentage of physical CPU that a virtual CPU of this logical system is guaranteed.

On AIX SPLPAR, this metric is equivalent to "Minimum Capacity" field of 'lparstat -i' command.

For WPARs, it is the minimum CPU share assigned to a WPAR that is guaranteed. WPAR shares CPU units of its global environment.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a host, the metric is equivalent to total number of cores on the host. For a resource pool and a logical system, this metrics indicates the guaranteed minimum CPU units configured for it.

On Solaris Zones, this metrics indicates the configured minimum CPU percentage reserved for a logical system.

For Solaris Zones, this metric is calculated as:

$$\text{BYLS_CPU_ENTL_MIN} = (\text{BYLS_CPU_SHARES_PRIO} / \text{Pool-Cpu-Shares})$$

where, Pool-Cpu-Shares is the total CPU shares available with CPU pool the zone is associated with. Pool-Cpu-Shares is addition of BYLS_CPU_SHARES_PRIO values for all active zones associated with this pool.

BYLS_CPU_ENTL_UTIL

Percentage of entitled processing units (guaranteed processing units allocated to this logical system) consumed by the logical system.

On a HP-UX HPVM host the metric indicates the logical system's CPU utilization with respect to minimum CPU entitlement.

On HP-UX HPVM host, this metric is calculated as: $BYLS_CPU_ENTL_UTIL = (BYLS_CPU_PHYSC / (BYLS_CPU_ENTL_MIN * BYLS_NUM_CPU)) * 100$

On AIX, this metric is calculated as: $BYLS_CPU_ENTL_UTIL = (BYLS_CPU_PHYSC / BYLS_CPU_ENTL) * 100$

On WPAR, this metric is calculated as: $BYLS_CPU_ENTL_UTIL = (BYLS_CPU_PHYSC / BYLS_CPU_ENTL_MAX) * 100$ This metric matches "%Resc" of topas command (inside WPAR)

On Solaris Zones, the metric indicates the logical system's CPU utilization with respect to minimum CPU entitlement. This metric is calculated as:

$BYLS_CPU_ENTL_UTIL = (BYLS_CPU_TOTAL_UTIL / BYLS_CPU_SHARES_PRIO) * 100$

If a Solaris zone is not assigned a CPU entitlement value then a CPU entitlement value is derived for this zone based on total CPU entitlement associated with the CPU pool this zone is attached to.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a host the value is same as $BYLS_CPU_PHYSC_TOTAL_UTIL$ while for logical system and resource pool the value is the percentage of processing units consumed w.r.t minimum CPU entitlement.

BYLS_CPU_MT_ENABLED

Indicates whether the CPU hardware threads are enabled("On") or not("Off") for a logical system. For AIX wpar, the metric will be "na".

On vMA, this metric indicates whether the CPU hardware threads are enabled or not for a host while for a resource pool and a logical system the value is not available("na").

BYLS_CPU_PHYSC

This metric indicates the number of CPU units utilized by the logical system.

On an Uncapped logical system, this value will be equal to the CPU units capacity used by the logical system during the interval. This can be more than the value entitled for a logical system.

BYLS_CPU_PHYS_READY_UTIL

On vMA, for a logical system it is the percentage of time, during the interval, that the CPU was in ready state. For a host and resource pool the value is NA.

BYLS_CPU_PHYS_SYS_MODE_UTIL

The percentage of time the physical CPUs were in system mode (kernel mode) for the logical system during the interval.

On AIX LPAR, this value is equivalent to "%sys" field reported by the "lparstat" command.

On Hyper-V host, this metric indicates the percentage of time spent in Hypervisor code.

On vMA, the metric indicates the percentage of time the physical CPUs were in system mode during the interval for the host or logical system. On vMA, for a resource pool, this metric is "na".

BYLS_CPU_PHYS_TOTAL_TIME

Total time in seconds, spent by the logical system on the physical CPUs.

On vMA, the value indicates the time spent in seconds on the physical CPU. by logical system or host or resource pool,

BYLS_CPU_PHYS_TOTAL_UTIL

Percentage of total time the physical CPUs were utilized by this logical system during the interval.

On vMA, the value indicates percentage of total time the physical CPUs were utilized by logical system or host or resource pool,

BYLS_CPU_PHYS_USER_MODE_UTIL

The percentage of time the physical CPUs were in user mode for the logical system during the interval.

On AIX LPAR, this value is equivalent to "%user" field reported by the "lparstat" command.

On Hyper-V host, this metric indicates the percentage of time spent in guest code.

On vMA, the metrics indicates the percentage of time the physical CPUs were in user mode during the interval for the host or logical system. On vMA, for a resource pool, this metric is "na".

BYLS_CPU_PHYS_WAIT_UTIL

On vMA, for a logical system it is the percentage of time, during the interval, that the virtual CPU was waiting for the IOs to complete. For a host and resource pool the value is NA.

BYLS_CPU_SHARES_PRIO

This metric indicates the weightage/priority assigned to a Uncapped logical system. This value determines the minimum share of unutilized processing units that this logical system can utilize.

On AIX SPLPAR this value is dependent on the available processing units in the pool and can range from 0 to 255.

For WPARs, this metric represents how much of a particular resource a WPAR receives relative to the other WPARs.

On vMA, for logical system and resource pool this value can range from 1 to 1000000 while for host the value is NA.

On Solaris Zones, this metric sets a limit on the number of fair share scheduler (FSS) CPU shares for a zone.

On Hyper-V host, this metric specifies allocation of CPU resources when more than one virtual machine is running and competing for resources. This value can range from 0 to 10000. For Root partition, this metric is NA.

BYLS_CPU_SYS_MODE_UTIL

On vMA, for a host and a logical system, this metric indicates the percentage of time the CPU was in system mode.

On vMA, for a resource pool, this metric is "na".

during the interval.

BYLS_CPU_TOTAL_UTIL

Percentage of total time the logical CPUs were not idle during this interval.

This metric is calculated against the number of logical CPUs configured for this logical system.

For AIX wpar, the metric represents the percentage of time the physical CPUs were not idle during this interval.

BYLS_CPU_UNRESERVED

On vMA, for host, it is the number of CPU cycles that are available for creating a new logical system. For a logical system and resource pool the value is NA.

BYLS_CPU_USER_MODE_UTIL

On vMA, for a host and a logical system, this metric indicates the percentage of time the CPU was in user mode during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DATACENTER_NAME

On vMA, for a host it is the name of the datacenter to which the host belongs to when it is managed by virtual center.

To uniquely identify datacenter in a virtual center, datacenter name is appended with the folder names in bottom up order.

For a logical system and resource pool, the value is NA.

BYLS_DISK_PHYS_BYTE

On vMA, for a host and a logical system, this metric indicates the number of KBs transferred to and from disks during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_BYTE_RATE

On vMA, for a host and a logical system, this metric indicates the average number of KBs per second at which data was transferred to and from disks during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_READ

On vMA, for a host and a logical system this metric indicates the number of physical reads during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_READ_BYTE_RATE

On vMA, for a host and a logical system, this metric indicates the average number of KBs transferred from the disk per second during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_READ_RATE

On vMA, for a host and a logical system, this metric indicates the number of physical reads per second during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_WRITE

On vMA, for a host and a logical system, this metric indicates the number of physical writes during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_WRITE_BYTE_RATE

On vMA, for a host and a logical system, this metric indicates the average number of KBs transferred to the disk per second during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_PHYS_WRITE_RATE

On vMA, for a host and a logical system, this metric indicates the number of physical writes per second during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_DISK_UTIL

On vMA, for a host, it is the average percentage of time during the interval (average utilization) that all the disks had IO in progress. For logical system and resource pool the value is NA.

BYLS_DISK_UTIL_PEAK

On vMA, for a host, it is the utilization of the busiest disk during the interval. For a logical system and resource pool the value is NA.

BYLS_DISPLAY_NAME

On vMA, this metric indicates the name of the host or logical system or resource pool.

On HPVM, this metric indicates the Virtual Machine name of the logical system and is equivalent to "Virtual Machine Name" field of 'hpvmstatus' command.

On AIX the value is as returned by the command "uname -n" (that is, the string returned from the "hostname" program).

On Solaris Zones, this metric indicates the zone name and is equivalent to 'NAME' field of 'zoneadm list -vc' command.

On Hyper-V host, this metric indicates the Virtual Machine name of the logical system and is equivalent to the Name displayed in Hyper-V Manager. For Root partition, the value is always "Root".

BYLS_IP_ADDRESS

This metric indicates IP Address of the particular logical system.

On vMA, this metric indicates the IP Address for a host and a logical system while for a resource pool the value is NA.

BYLS_LS_HOSTNAME

This is the DNS registered name of the system.

On Hyper-V host, this metric is NA if the logical system is not active or Hyper-V Integration Components are not installed on it.

On vMA, for a host and logical system the metric is the Fully Qualified Domain Name, while for resource pool the value is NA.

BYLS_LS_HOST_HOSTNAME

On vMA, for logical system and resource pool, it is the FQDN of the host on which they are hosted. For a host, the value is NA.

BYLS_LS_ID

An unique identifier of the logical system.

On HPVM, this metric is a numeric id and is equivalent to "VM # " field of 'hvvmstatus' command.

On AIX LPAR, this metric indicates partition number and is equivalent to "Partition Number" field of 'lparstat -i' command. For aix wpar, this metric represents the partition number and is equivalent to "uname -W" from inside wpar.

On Solaris Zones, this metric indicates the zone id and is equivalent to 'ID' field of 'zoneadm list -vc' command.

On Hyper-V host, this metric indicates the PID of the process corresponding to this logical system. For Root partition, this metric is NA.

On vMA, this metric is a unique identifier for a host, resource pool and a logical system. The value of this metric may change for an instance across collection intervals.

BYLS_LS_MODE

This metric indicates whether the CPU entitlement for the logical system is Capped or Uncapped.

On AIX SPLPAR, this metric is same as "Mode" field of 'lparstat -i' command.

For WPARs, this metric is always CAPPED.

On vMA, the value is Capped for a host and Uncapped for a logical system. For resource pool, the value is Uncapped or Capped depending on whether the reservation is expandable or not for it.

On Solaris Zones, this metric is "Capped" when the zone is assigned CPU shares and is attached to a valid CPU pool.

BYLS_LS_NAME

This is the name of the computer.

On HPVM, this metric indicates the Virtual Machine name of the logical system and is equivalent to "Virtual Machine Name" field of 'hvvmstatus' command.

On AIX the value is as returned by the command "uname -n" (that is, the string returned from the "hostname" program).

On vMA, this metric is a unique identifier for host, resource pool and a logical system. The value of this metric remains the same, for an instance, across collection intervals.

On Solaris Zones, this metric indicates the zone name and is equivalent to 'NAME' field of 'zoneadm list -vc' command.

On Hyper-V host, this metric indicates the name of the XML file which has configuration information of the logical system. This file will be present under the logical system's installation directory indicated by BYLS_LS_PATH. For Root partition, the value is always "Root".

BYLS_LS_OSTYPE

The Guest OS this logical system is hosting.

On HPVM, the metric can have following values: HP-UX Linux Windows OpenVMS Other Unknown

On Hyper-V host, the metric can have following values: Windows Other

On Hyper-V host, this metric is NA if the logical system is not active or Hyper-V Integration Components are not installed on it.

On vMA, the metric can have the following values for host and logical system: ESX-Serv (applicable only for a host) Linux Windows Solaris Unknown The value is NA for resource pool

BYLS_LS_PARENT_TYPE

On vMA, the metric indicates the type of parent entity. The value is HOST if the parent is a host, RESPOOL if the parent is resource pool. For a host, the value is NA.

BYLS_LS_PARENT_UUID

On vMA, the metric indicates the UUID of the parent entity. For logical system and resource pool this metric could indicate the UUID of a host or resource pool as they can be created under a host or resource pool. For a host, the value is NA.

BYLS_LS_PATH

This metric indicates the installation path for the logical system.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, the metric indicates the installation path for host or logical system. On vMA, for a resource pool and a host, this metric is "na".

BYLS_LS_ROLE

On vMA, for a host the metric is HOST. For a logical system the value is GUEST and for a resource pool the value is RESPOOL. For logical system which is a vMA, the value is PROXY.

BYLS_LS_SHARED

This metric indicates whether the physical CPUs are dedicated to this logical system or shared.

On HPVM, and Hyper-V host, this metric is always "Shared".

On vMA, the value is "Dedicated" for host, and "Shared" for logical system and resource pool.

On AIX SPLPAR, this metric is equivalent to "Type" field of 'lparstat -i' command. For AIX wpar, this metric will be always "Shared".

On Solaris Zones, this metric is "Dedicated" when this zone is attached to a CPU pool not shared by any other zone.

BYLS_LS_STATE

The state of this logical system.

On HPVM, the logical systems can have one of the following states: Unknown Other invalid Up Down Boot Crash Shutdown Hung

On vMA, this metric can have one of the following states for a host: on off suspended The value is NA for resource pool.

On Solaris Zones, the logical systems can have one of the following states: configured incomplete installed ready running shutting down mounted

On AIX lpars, the logical system will be always active. On AIX wpar, the logical systems can have one of the following states: Broken Transitional Defined Active Loaded Paused Frozen Error

A logical system on a Hyper-V host can have the following states: unknown enabled disabled paused suspended starting snapshotting migrating saving stopping deleted pausing resuming

BYLS_LS_TYPE

The type of this logical system. On AIX, the logical systems can have one of the following types: lpar sys wpar app wpar

On vMA, the value of this metric is "VMware".

BYLS_LS_UUID

UUID of this logical system. This Id uniquely identifies this logical system across multiple hosts.

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for a logical system or a host, the value indicates the UUID of the system. For a resource pool the value is hostname of the host where resource pool is hosted followed by the unique id of resource pool.

BYLS_MACHINE_MODEL

On vMA, for a host, it is the CPU model of the host system. For a logical system and resource pool the value is "na".

BYLS_MEM_ACTIVE

On vMA, for a logical system it is the amount of memory, that is actively used. For a host and resource pool the value is NA.

BYLS_MEM_AVAIL

On vMA, for a host, the amount of physical available memory in the host system (in MBs unless otherwise specified). For a logical system and resource pool the value is NA.

BYLS_MEM_BALLOON_USED

On vMA, for logical system, it is the amount of memory held by memory control for ballooning. The value is represented in KB. For a host and resource pool the value is NA.

BYLS_MEM_BALLOON_UTIL

On vMA, for logical system, it is the amount of memory held by memory control for ballooning. It is represented as a percentage of BYLS_MEM_ENTL. For a host, and resource pool the value is NA.

BYLS_MEM_ENTL

The minimum memory configured for this logical system (in MB).

On Hyper-V host, for Root partition, this metric is NA.

On vMA, for host the value is the physical memory available in the system and for logical system this metric indicates the minimum memory configured while for resource pool the value is NA.

BYLS_MEM_ENTL_MAX

In a virtual environment, this metric indicates the maximum amount of memory configured for a logical system (in MB). The value of this metric will be "-3" in PA and "ul" in other clients if entitlement is 'Unlimited' for a logical system. On AIX LPARs, this metric will be "na".

On vMA, this metric indicates the maximum amount of memory configured, in MB, for resource pool and a logical system. For a host, the value is the amount of physical memory available in the system.

BYLS_MEM_ENTL_MIN

In a virtual environment, this metric indicates the minimum amount of memory configured for a logical system (in MB). On AIX LPARs, this metric will be "na".

On vMA, this metric indicates the reserved amount of memory configured, in MB, for a host, resource pool and a logical system.

BYLS_MEM_ENTL_UTIL

The percentage of entitled memory in use during the interval. This includes system memory (occupied by the kernel), buffer cache and user memory.

On vMA, for a logical system or a host, the value indicates percentage of entitled memory in use during the interval by it. On vMA, for a resource pool, this metric is "na".

BYLS_MEM_FREE

On vMA, for a host and logical system, it is the amount of memory not allocated (in MBs unless otherwise specified). For a resource pool the value is NA.

BYLS_MEM_FREE_UTIL

On vMA, for a host and logical system, it is the percentage of physical memory that was free at the end of the interval. For a resource pool the value is NA.

BYLS_MEM_HEALTH

On vMA, for a host, it is a number that indicates the state of the memory. Low number indicates system is not under memory pressure. For a logical system and resource pool the value is "na".

The possible free memory thresholds that are applicable are :

0 - High - indicates free memory is available and no memory pressure. 1 - Soft 2 - Hard 3 - Low - indicates there is a pressure for free memory.

BYLS_MEM_OVERHEAD

The amount of memory associated with a logical system, that is currently consumed on the host system, due to virtualization.

On vMA, this metric indicates the amount of overhead memory associated with a host, logical system and resource pool.

BYLS_MEM_PHYS

On vMA, for host the value is the physical memory available in the system and for logical system this metric indicates the minimum memory configured. On vMA, for a resource pool, this metric is "na".

BYLS_MEM_PHYS_UTIL

On vMA, the metric indicates the percentage of physical memory used during the interval by a host, logical system. On vMA, for a resource pool, this metric is "na".

BYLS_MEM_SHARES_PRIO

The weightage/priority for memory assigned to this logical system. This value influences the share of unutilized physical Memory that this logical system can utilize. On AIX LPARs, this metric will be "na".

On vMA, this metric indicates the share of memory configured to a resource pool and a logical system. For a host the value is NA.

BYLS_MEM_SWAPIN

On vMA, for a logical system the value indicates the amount of memory that is swapped in during the interval. For a host and resource pool the value is NA.

BYLS_MEM_SWAPOUT

On vMA, for a logical system the value indicates the amount of memory that is swapped out during the interval. For a host and resource pool the value is NA.

BYLS_MEM_SWAPPED

On vMA, for a host, logical system and resource pool, this metrics indicates the amount of memory that has been transparently swapped to and from the disk.

BYLS_MEM_SWAPTARGET

On vMA, for a logical system the value indicates the amount of memory that can be swapped. For a host and resource pool the value is "na".

BYLS_MEM_SWAP_UTIL

On Solaris, this metric indicates the percentage of swap memory consumed by the zone with respect to total configured swap memory (BYLS_MEM_SWAP). This metric is calculated as : $BYLS_MEM_SWAP_UTIL = (BYLS_MEM_SWAP_USED) / (BYLS_MEM_SWAP) * 100$

On vMA, for a logical system, it is the percentage of swap memory utilized w.r.t the amount of swap memory available for a logical system. For host and resource pool the value is NA. For a logical system this metric is calculated using the below formula: $(BYLS_MEM_SWAPPED * 100) / (BYLS_MEM_ENTL - BYLS_MEM_ENTL_MIN)$

BYLS_MEM_SYS

On vMA, for a host, it is the amount of physical memory (in MBs unless otherwise specified) used by the system (kernel) during the interval. For logical system and resource pool the value is NA.

BYLS_MEM_UNRESERVED

On vMA, for a host it is the amount of memory, that is unreserved. For a logical system and resource pool the value is "na".

Memory reservation not used by the Service Console, VMkernel, vSphere services and other powered on VMs user-specified memory reservations and overhead memory.

BYLS_MEM_USED

On vMA, for host, resource pool and logical system the value indicates the amount of memory used represented in MB. On vMA, for a resource pool, this metric is "na".

BYLS_NET_BYTE_RATE

On vMA, for a host and logical system, it is the sum of data transmitted and received for all the NIC instances of the host and virtual machine. It is represented in KBps. For a resource pool the value is NA.

BYLS_NET_IN_BYTE

On vMA, for a host and logical system, it is number of bytes, in MB, received during the interval. For a resource pool the value is NA.

BYLS_NET_IN_PACKET

On vMA, for a host and a logical system, this metric indicates the number of successful packets received through all network interfaces during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_NET_IN_PACKET_RATE

On vMA, for a host and a logical system, this metric indicates the number of successful packets per second received through all network interfaces during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_NET_OUT_BYTE

On vMA, for a host and logical system, it is the number of bytes, in MB, transmitted during the interval. For a resource pool the value is NA.

BYLS_NET_OUT_PACKET

On vMA, for a host and a logical system, it is the number of successful packets sent through all network interfaces during the last interval. On vMA, for a resource pool, this metric is "na".

BYLS_NET_OUT_PACKET_RATE

On vMA, for a host and a logical system, this metric indicates the number of successful packets per second sent through the network interfaces during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_NET_PACKET_RATE

On vMA, for a host and a logical system, it is the number of successful packets per second, both sent and received, for all network interfaces during the interval. On vMA, for a resource pool, this metric is "na".

BYLS_NUM_ACTIVE_LS

On vMA, for a host, this indicates the number of logical systems hosted in a system that are active. For a logical system and resource pool the value is NA.

BYLS_NUM_CPU

The number of virtual CPUs configured for this logical system. This metric is equivalent to GBL_NUM_CPU on the corresponding logical system.

On HPVM, the maximum CPUs a logical system can have is 4 with respect to HPVM 3.x.

On AIX SPLPAR, the number of CPUs can be configured irrespective of the available physical CPUs in the pool this logical system belongs to. For AIX wpars, this metric represents the logical CPUs of the global environment.

On vMA, for a host the metric is the number of physical CPU threads on the host. For a logical system, the metric is the number of virtual cpus configured. For a resource pool the metric is NA.

On Solaris Zones, this metric represents number of CPUs in the CPU pool this zone is attached to. This metric value is equivalent to GBL_NUM_CPU inside corresponding non-global zone.

BYLS_NUM_CPU_CORE

On vMA, for a host this metric provides the total number of CPU cores on the system. For a logical system or a resource pool the value is NA.

BYLS_NUM_DISK

The number of disks configured for this logical system. Only local disk devices and optical devices present on the system are counted in this metric.

On vMA, for a host the metric is the number of disks configured for the host. For a logical system, the metric is the number of logical disk devices present on the logical system. For a resource pool the metric is NA.

For AIX wpars, this metric will be "na".

On Hyper-V host, this metric value is equivalent to GBL_NUM_DISK inside corresponding Hyper-V guest.

On Hyper-V host, this metric is NA if the logical system is not active.

BYLS_NUM_LS

On vMA, for a host, this indicates the number of logical systems hosted in a system. For a logical system and resource pool the value is NA.

BYLS_NUM_NETIF

The number of network interfaces configured for this logical system.

On LPAR, this metric includes the loopback interface.

On Hyper-V host, this metric value is equivalent to GBL_NUM_NETWORK inside corresponding Hyper-V guest.

On Solaris Zones, this metric value is equivalent to GBL_NUM_NETWORK inside corresponding non-global zone.

On Hyper-V host, this metric is NA if the logical system is not active.

On vMA, for a host the metric is the number of network adapters on the host. For a logical system, the metric is the number of network interfaces configured for the logical system. For a resource pool the metric is NA.

BYLS_NUM_SOCKET

On vMA, for a host, this metrics indicates the number of physical cpu sockets on the system. For a logical system or a resource pool the value is NA.

BYLS_UPTIME_HOURS

On vMA, for a host and logical system the metrics is the time, in hours, since the last system reboot. For a resource pool the value is NA.

BYLS_UPTIME_SECONDS

The uptime of this logical system in seconds.

On AIX LPARs, this metric will be "na".

On vMA, for a host and logical system the metric is the uptime in seconds while for a resource pool the metric is NA.

BYLS_VC_IP_ADDRESS

On vMA, for a host, the metric indicates the IP address of the Virtual Centre that the host is managed by. For a resource pool and logical system the value is NA.

BYNETIF_COLLISION

The number of physical collisions that occurred on the network interface during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

For HP-UX, this will be the same as the sum of the "Single Collision Frames", "Multiple Collision Frames", "Late Collisions", and "Excessive Collisions" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For most other Unix systems, this is the same as the sum of the "Coll" column from the "netstat -i" command ("collisions" from the "netstat -i -e" command on Linux) for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_COLLISION_1_MIN_RATE

The number of physical collisions per minute on the network interface during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_COLLISION_RATE

The number of physical collisions per second on the network interface during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_COLLISION_RATE_CUM

The average number of physical collisions per second on the network interface over the cumulative collection time. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not currently include deferred packets.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

This data is not collected for non-broadcasting devices, such as loopback (lo), and is always zero.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_ERROR

The number of physical errors that occurred on the network interface during the interval. An increasing number of errors may indicate a hardware problem in the network.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

For HP-UX, this will be the same as the sum of the "Inbound Errors" and "Outbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "lerrs" (RX-ERR on Linux) and "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_ERROR_1_MIN_RATE

The number of physical errors per minute on the network interface during the interval.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_ERROR_RATE

The number of physical errors per second on the network interface during the interval.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

BYNETIF_ERROR_RATE_CUM

The average number of physical errors per second on the network interface over the cumulative collection time.

On Unix systems, this data is not available for loop-back (lo) devices and is always zero.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric will be N/A.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_ID

The ID number of the network interface.

BYNETIF_IN_BYTE

The number of KBs received from the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_BYTE_RATE

The number of KBs per second received from the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_BYTE_RATE_CUM

The average number of KBs per second received from the network via this interface over the cumulative collection time. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_PACKET

The number of successful physical packets received through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

For HP-UX, this will be the same as the sum of the "Inbound Unicast Packets" and "Inbound Non-Unicast Packets" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Ipkts" column (RX-OK on Linux) from the "netstat -i" command for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_PACKET_RATE

The number of successful physical packets per second received through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface

(127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_IN_PACKET_RATE_CUM

The average number of physical packets per second received through the network interface over the cumulative collection time.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_NAME

The name of the network interface.

For HP-UX 11.0 and beyond, these are the same names that appear in the "Description" field of the "lanadmin" command output.

On all other Unix systems, these are the same names that appear in the "Name" column of the "netstat -i" command.

Some examples of device names are:

```
lo - loop-back driver
ln - Standard Ethernet driver
en - Standard Ethernet driver
le - Lance Ethernet driver
ie - Intel Ethernet driver
tr - Token-Ring driver
et - Ether Twist driver
bf - fiber optic driver
```

All of the device names will have the unit number appended to the name. For example, a loop-back device in unit 0 will be "lo0".

On vMA for Lan cards which are of type ESXVLan, this metric contains the vmnic<number> as first half and the second half is the ESX host name.

BYNETIF_NET_TYPE

The type of network device the interface communicates through.

```
Lan      - local area network card
Loop     - software loopback
          interface (not tied to a
          hardware device)
Loop6    - software loopback
          interface IPv6 (not tied
          to a hardware device)
Serial   - serial modem port
Vlan     - virtual lan
```

Wan - wide area network card
Tunnel - tunnel interface
Apa - HP LinkAggregate Interface (APA)
Other - hardware network interface
type is unknown.
ESXVLan - The card type belongs to network cards of ESX hosts which are monitored on vMA.

BYNETIF_OUT_BYTE

The number of KBs sent to the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_BYTE_RATE

The number of KBs per second sent to the network via this interface during the interval. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_BYTE_RATE_CUM

The average number of KBs per second sent to the network via this interface over the cumulative collection time. Only the bytes in packets that carry data are included in this rate.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_PACKET

The number of successful physical packets sent through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

For HP-UX, this will be the same as the sum of the "Outbound Unicast Packets" and "Outbound Non-Unicast Packets" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Opkts" column (TX-OK on Linux) from the "netstat -i" command for a network device. See also netstat(1).

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_PACKET_RATE

The number of successful physical packets per second sent through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_OUT_PACKET_RATE_CUM

The average number of successful physical packets per second sent through the network interface over the cumulative collection time.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYNETIF_PACKET_RATE

The number of successful physical packets per second sent and received through the network interface during the interval. Successful packets are those that have been processed without errors or collisions.

If BYNETIF_NET_TYPE is "ESXVLan", then this metric shows the values for the Lan card in the host.

Physical statistics are packets recorded by the network drivers. These numbers most likely will not be the same as the logical statistics. The values returned for the loopback interface will show "na" for the physical statistics since there is no network driver activity.

Logical statistics are packets seen only by the Interface Protocol (IP) layer of the networking subsystem. Not all packets seen by IP will go out and come in through a network driver. An example is the loopback interface (127.0.0.1). Pings or other network generating commands (ftp, rlogin, and so forth) to 127.0.0.1 will not change physical driver statistics. Pings to IP addresses on remote systems will change physical driver statistics.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

BYOP_CLIENT_COUNT

The number of current NFS operations that the local machine has processed as a NFS client during the interval.

A host on the network can act both as a client, or as a server at the same time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYOP_CLIENT_COUNT_CUM

The number of current NFS operations that the local machine has processed as a NFS client over the cumulative collection time.

A host on the network can act both as a client, or as a server at the same time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYOP_INTERVAL

The amount of time in the interval.

BYOP_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYOP_NAME

String mnemonic for the NFS operation. One of the following:

For NFS Version 2

Name	Operation/Action
getattr	Return the current attributes of a file.
setattr	Set the attributes of a file and returns the new attributes.
lookup	Return the attributes of a file.
readlink	Return the string in the symbolic link of a file.
read	Return data from a file.
write	Put data into a file.
create	Create a file.
remove	Remove a file.
rename	Give a file a new name.
link	Create a hard link to a file.
symlink	Create a symbolic link to a file.

mkdir	Create a directory.
rmdir	Remove a directory.
readdir	Read a directory entry.
statfs	Return mounted file system information.
null	Verify NFS service connections and timing. On HP-UX, no actual work done.
writocache	Flush the server write cache if a special write cache exists. Most systems use the file buffer cache and not a special server cache. Not used on HP-UX.
root	Find root file system handle (probably obsolete). Not used on HP-UX.

For NFS Version 3

Name	Operation/Action
getattr	Return the current attributes of a file.
setattr	Set the attributes of a file and returns the new attributes.
lookup	Return the attributes of a file.
access	Check access permissions of a user.
readlink	Return the string in the symbolic link of a file.
read	Return data from a file.
write	Put data into a file.
create	Create a file.
mkdir	Make a directory.
symlink	Create a symbolic link to a file.
mknod	Create a special device.
remove	Remove a file.
rmdir	Remove a directory.
rename	Give a file a new name.
link	Create a hard link to a file.
readdir	Read a directory entry.

readdirplus	Extended read of a directory entry.
fsstat	Get dynamic file system information.
fsinfo	Get static file system information.
pathconf	Retrieve POSIX information.
commit	Commit cached data on server to stable storage.
null	Verify NFS services. No actual work done.

BYOP_SERVER_COUNT

The number of current NFS operations that the local machine has processed as a NFS server during the interval. A host on the network can act both as a client, or as a server at the same time.

BYOP_SERVER_COUNT_CUM

The number of current NFS operations that the local machine has processed as a NFS server over the cumulative collection time.

A host on the network can act both as a client, or as a server at the same time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

BYSWP_SWAP_PRI

The priority of this swap device. This value is set by either the swapon(1M) command, or by the "pri=" field in /etc/fstab.

On HP-UX, swap space is used by the lower value priorities first. Since device swap is faster than file system swap, it is advisable to have lower values for device swap. The legal values for priority range from 0 to 10.

On HP-UX, the "memory" swap area has no priority and will be shown as -1. This indicates that using memory as a swap area is only done after all other swap resources have been exhausted. This is true in extreme cases of memory pressure forcing the kernel to swap the entire process to disk. In cases of process deactivation, the memory pseudo swap actually has the highest priority - deactivated pages are not moved - they are simply marked as deactivated and the space they occupy is considered pseudo swap.

On Linux, swap space is used by the higher value priorities first. The legal values for priority range from 0 to 32767. The system assigns negative priority values if no priority is specified during the creation of swap area. See swapon(8) for details.

BYSWP_SWAP_SPACE_AVAIL

The capacity (in MB) for swapping in this swap area.

On HP-UX, for "device" type swap, this value is constant. However, for "filesystem" swap this value grows as needed. File system swap grows in units of "SWCHUNKS" x DEV_BSIZE bytes, which is typically 2MB. This metric is similar to the "AVAIL" parameters returned from /usr/sbin/swapinfo. For "memory" type swap, this value also grows as needed or as possible, given that any memory reserved for swap cannot be used for normal virtual memory. Note that this is potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space.

On SUN, this is the same as (blocks * .5)/1024, reported by the "swap -l" command.

On AIX, this metric is set to "na" for inactive swap devices.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

BYSWP_SWAP_SPACE_NAME

On Unix systems, this is the name of the device file or file system where the swap space is located.

On HP-UX, part of the system's physical memory may be allocated as a pseudo-swap device. It is enabled by setting the "SWAPMEM_ON" kernel parameter to 1.

On SunOS 5.X, part of the system's physical memory may be allocated as a pseudo-swap device. Also note, "/tmp" is usually configured as a memory based file system and is not used for swap space. Therefore, it will not be listed with the swap devices. This is noted because "df" uses the label "swap" for the "/tmp" file system which may be confusing. See tmpfs(7).

BYSWP_SWAP_SPACE_USED

The amount of swap space (in MB) used in this area.

On HP-UX, this value is similar to the "USED" column returned by the /usr/sbin/swapinfo command.

On SUN, "Used" indicates amount written to disk (or locked in memory), rather than reserved. Swap space is reserved (by decrementing a counter) when virtual memory for a program is created. This is the same as (blocks - free) * .5/1024, reported by the "swap -l" command.

On SUN, global swap space is tracked through the operating system. Device swap space is tracked through the devices. For this reason, the amount of swap space used may differ between the global and by-device metrics. Sometimes pages that are marked to be swapped to disk by the operating system are never swapped. The operating system records this as used swap space, but the devices do not, since no physical IOs occur. (Metrics with the prefix "GBL" are global and metrics with the prefix "BYSWP" are by device.)

On AIX, this metric is set to "na" for inactive swap devices.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

BYSWP_SWAP_TYPE

The type of swap space allocated on the system.

On HP-UX and SUN, types of swap space are device, file system ("filesys"), or memory. "Device" swap is accessed directly without going through the file system, and is therefore faster than "filesys" swap. "Filesys" swap can be to a local or NFS mounted swap file. "Memory" swap is space in the system's physical memory reserved for pseudo-swap for running processes. Using pseudo-swap means the pages are simply locked in memory rather than copied to a swap area.

On SUN, note that "/tmp" is usually configured as a memory based file system and is not used for swap space. Therefore, it will not be listed with the swap devices, and "swap" or "tmpfs" will not be swap types. This is noted because "df" uses the label "swap" for the "/tmp" file system which may be confusing. See tmpfs(7).

On AIX, "Device" swap is accessed directly without going through the file system. For "Device" swap, the device is specially allocated for swapping purpose only. The device can be logical volume, "lv" or remote file system, "remote fs". The swap is often referred as paging to paging space.

FS_BLOCK_SIZE

The maximum block size of this file system, in bytes.

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_DEVNAME

On Unix systems, this is the path name string of the current device.

On Windows, this is the disk drive string of the current device.

On HP-UX, this is the "fsname" parameter in the mount(1M) command. For NFS devices, this includes the name of the node exporting the file system. It is possible that a process may mount a device using the mount(2) system call. This call does not update the "/etc/mnttab" and its name is blank. This situation is rare, and should be corrected by syncer(1M). Note that once a device is mounted, its entry is displayed, even after the device is unmounted, until the mdaemon process terminates.

On SUN, this is the path name string of the current device, or "tmpfs" for memory based file systems. See tmpfs(7).

FS_DEVNO

On Unix systems, this is the major and minor number of the file system.

On Windows, this is the unit number of the disk device on which the logical disk resides.

FS_DIRNAME

On Unix systems, this is the path name of the mount point of the file system.

On Windows, this is the drive letter associated with the selected disk partition.

On HP-UX, this is the path name of the mount point of the file system if the logical volume has a mounted file system. This is the directory parameter of the mount(1M) command for most entries. Exceptions are:

- * For lvm swap areas, this field contains "lvm swap device".
- * For logical volumes with no mounted file systems, this field contains "Raw Logical Volume" (relevant only to Perf Agent).

On HP-UX, the file names are in the same order as shown in the "/usr/sbin/mount -p" command. File systems are not displayed until they exhibit IO activity once the mdaemon has been started. Also, once a device is displayed, it continues to be displayed (even after the device is unmounted) until the mdaemon process terminates.

On SUN, only "UFS", "HSFS" and "TMPFS" file systems are listed. See mount(1M) and mnttab(4). "TMPFS" file systems are memory based filesystems and are listed here for convenience. See tmpfs(7).

On AIX, see mount(1M) and filesystems(4). On OSF1, see mount(2).

FS_FRAG_SIZE

The fundamental file system block size, in bytes.

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_INODE_UTIL

Percentage of this file system's inodes in use during the interval.

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_MAX_INODES

Number of configured file system inodes.

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

FS_MAX_SIZE

Maximum number that this file system could obtain if full, in MB.

Note that this is the user space capacity - it is the file system space accessible to non root users. On most Unix systems, the df command shows the total file system capacity which includes the extra file system space accessible to root users only.

The equivalent fields to look at are "used" and "avail". For the target file system, to calculate the maximum size in MB, use

$$\text{FS Max Size} = (\text{used} + \text{avail})/1024$$

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

On HP-UX, this metric is updated at 4 minute intervals to minimize collection overhead.

FS_PHYS_IO_RATE

The number of physical IOs per second directed to this file system during the interval.

FS_PHYS_IO_RATE_CUM

The average number of physical IOs per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

FS_PHYS_READ_BYTE_RATE

The number of physical KBs per second read from this file system during the interval.

FS_PHYS_READ_BYTE_RATE_CUM

The average number of KBs per second of physical reads from this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

FS_PHYS_READ_RATE

The number of physical reads per second directed to this file system during the interval.

On Unix systems, physical reads are generated by user file access, virtual memory access (paging), file system management, or raw device access.

FS_PHYS_READ_RATE_CUM

The average number of physical reads per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

FS_PHYS_WRITE_BYTE_RATE

The number of physical KBs per second written to this file system during the interval.

FS_PHYS_WRITE_BYTE_RATE_CUM

The average number of KBs per second of physical writes to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

FS_PHYS_WRITE_RATE

The number of physical writes per second directed to this file system during the interval.

FS_PHYS_WRITE_RATE_CUM

The average number of physical writes per second directed to this file system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

FS_SPACE_RESERVED

The amount of file system space in MBs reserved for superuser allocation.

On AIX, this metric is typically zero because by default AIX does not reserve any file system space for the superuser.

FS_SPACE_USED

The amount of file system space in MBs that is being used.

FS_SPACE_UTIL

Percentage of the file system space in use during the interval.

Note that this is the user space capacity - it is the file system space accessible to non root users. On most Unix systems, the df command shows the total file system capacity which includes the extra file system space accessible to root users only.

A value of "na" may be displayed if the file system is not mounted. If the product is restarted, these unmounted file systems are not displayed until remounted.

On HP-UX, this metric is updated at 4 minute intervals to minimize collection overhead.

FS_TYPE

A string indicating the file system type. On Unix systems, some of the possible types are:

```
hfs   - user file system
ufs   - user file system
ext2  - user file system
cdfs  - CD-ROM file system
vxfs  - Veritas (vxfs) file system
nfs   - network file system
nfs3  - network file system
       Version 3
```

On Windows, some of the possible types are:

```
NTFS  - New Technology File System
FAT   - 16-bit File Allocation
       Table
FAT32 - 32-bit File Allocation
       Table
```

FAT uses a 16-bit file allocation table entry (216 clusters).

FAT32 uses a 32-bit file allocation table entry. However, Windows 2000 reserves the first 4 bits of a FAT32 file allocation table entry, which means FAT32 has a theoretical maximum of 228 clusters. NTFS is native file system of Windows NT and beyond.

GBL_ACTIVE_CPU

The number of CPUs online on the system.

For HP-UX and certain versions of Linux, the sar(1M) command allows you to check the status of the system CPUs.

For SUN and DEC, the commands psrinfo(1M) and psradm(1M) allow you to check or change the status of the system CPUs.

For AIX, the pstat(1) command allows you to check the status of the system CPUs.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment if RSET is not configured for the System WPAR. If RSET is configured for the System WPAR, this metric value will report the number of CPUs in the RSET.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_ACTIVE_CPU_CORE

This metric provides the total number of active CPU cores on a physical system.

GBL_ACTIVE_PROC

An active process is one that exists and consumes some CPU time. GBL_ACTIVE_PROC is the sum of the alive-process-time/interval-time ratios of every process that is active (uses any CPU time) during an interval.

The following diagram of a four second interval during which two processes exist on the system should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

	Seconds			
	1	2	3	4
Proc				
A	live	live	live	live
B	live/CPU	live/CPU	live	dead

Process A is alive for the entire four second interval but consumes no CPU. A's contribution to GBL_ALIVE_PROC is $4 \times \frac{1}{4}$. A contributes $0 \times \frac{1}{4}$ to GBL_ACTIVE_PROC. B's contribution to GBL_ALIVE_PROC is $3 \times \frac{1}{4}$. B contributes $2 \times \frac{1}{4}$ to GBL_ACTIVE_PROC. Thus, for this interval, GBL_ACTIVE_PROC equals 0.5 and GBL_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, GBL_ACTIVE_PROC will always be less than or equal to GBL_ALIVE_PROC.

This metric is a good overall indicator of the workload of the system. An unusually large number of active processes could indicate a CPU bottleneck.

To determine if the CPU is a bottleneck, compare this metric with GBL_CPU_TOTAL_UTIL and GBL_RUN_QUEUE. If GBL_CPU_TOTAL_UTIL is near 100 percent and GBL_RUN_QUEUE is greater than one, there is a bottleneck.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

GBL_ALIVE_PROC

An alive process is one that exists on the system. GBL_ALIVE_PROC is the sum of the alive-process-time/interval-time ratios for every process.

The following diagram of a four second interval during which two processes exist on the system should be used to understand the above definition. Note the difference between active processes, which consume CPU time, and alive processes which merely exist on the system.

	Seconds			
	1	2	3	4
Proc				
A	live	live	live	live
B	live/CPU	live/CPU	live	dead

Process A is alive for the entire four second interval but consumes no CPU. A's contribution to GBL_ALIVE_PROC is $4 \times \frac{1}{4}$. A contributes $0 \times \frac{1}{4}$ to GBL_ACTIVE_PROC. B's contribution to GBL_ALIVE_PROC is $3 \times \frac{1}{4}$. B contributes $2 \times \frac{1}{4}$ to GBL_ACTIVE_PROC. Thus, for this interval, GBL_ACTIVE_PROC equals 0.5 and GBL_ALIVE_PROC equals 1.75.

Because a process may be alive but not active, GBL_ACTIVE_PROC will always be less than or equal to GBL_ALIVE_PROC.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

GBL_BLANK

A string of blanks.

GBL_BOOT_TIME

The date and time when the system was last booted.

GBL_COLLECTOR

ASCII field containing collector name and version. The collector name will appear as either "SCOPE/xx V.UU.FF.LF" or "Coda RV.UU.FF.LF". xx identifies the platform; V = version, UU = update level, FF = fix level, and LF = lab fix id. For example, SCOPE/UX C.04.00.00; or Coda A.07.10.04.

GBL_COMPLETED_PROC

The number of processes that terminated during the interval.

On non HP-UX systems, this metric is derived from sampled process data. Since the data for a process is not available after the process has died on this operating system, a process whose life is shorter than the sampling interval may not be seen when the samples are taken. Thus this metric may be slightly less than the actual value. Increasing the sampling frequency captures a more accurate count, but the overhead of collection may also rise.

GBL_CPU_CLOCK

The clock speed of the CPUs in MHz if all of the processors have the same clock speed. Otherwise, "na" is shown if the processors have different clock speeds. Note that Linux supports dynamic frequency scaling and if it is enabled then there can be a change in CPU speed with varying load.

GBL_CPU_CYCLE_ENTL_MAX

On a recognized VMware ESX guest, where VMware guest SDK is enabled,, this value indicates the maximum processor capacity, in MHz, configured for this logical system. The value is -3 if entitlement is 'Unlimited' for this logical system.

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na".

On a standalone system, the value is the sum of clock speed of individual CPUs.

GBL_CPU_CYCLE_ENTL_MIN

On a recognized VMware ESX guest, where VMware guest SDK is enabled,, this value indicates the minimum processor capacity, in MHz, configured for this logical system.

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na".

On a standalone system, the value is the sum of clock speed of individual CPUs.

GBL_CPU_ENTL_MAX

In a virtual environment, this metric indicates the maximum number of processing units configured for this logical system.

On AIX SPLPAR, this metric is equivalent to "Maximum Capacity" field of 'lparstat -i' command.

On a recognized VMware ESX guest the value is equivalent to GBL_CPU_CYCLE_ENTL_MAX represented in CPU units.

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na".

On a standalone system the value is same as GBL_NUM_CPU.

GBL_CPU_ENTL_MIN

In a virtual environment, this metric indicates the minimum number of processing units configured for this Logical system.

On AIX SPLPAR, this metric is equivalent to "Minimum Capacity" field of 'lparstat -i' command.

On a recognized VMware ESX guest, where VMware guest SDK is enabled, the value is equivalent to GBL_CPU_CYCLE_ENTL_MIN represented in CPU units.

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na".

On a standalone system the value is same as GBL_NUM_CPU.

GBL_CPU_ENTL_UTIL

Percentage of entitled processing units (guaranteed processing units allocated to this logical system) consumed by the logical system.

On AIX, this metric is calculated as:

$$\text{GBL_CPU_ENTL_UTIL} = (\text{GBL_CPU_PHYSC} / \text{GBL_CPU_ENTL}) * 100$$

On a recognized VMware ESX guest, where VMware guest SDK is enabled, this metric is calculated as:

$$\text{GBL_CPU_ENTL_UTIL} = (\text{GBL_CPU_PHYSC} / \text{GBL_CPU_ENTL_MIN}) * 100$$

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na".

On a standalone system, the value is same as GBL_CPU_TOTAL_UTIL.

GBL_CPU_IDLE_TIME

The time, in seconds, that the CPU was idle during the interval. This is the total idle time, including waiting for I/O.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Solaris non-global zones, this metric is N/A. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_TIME_CUM

The time, in seconds, that the CPU was idle over the cumulative collection time. This is the total idle time, including waiting for I/O.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HPUX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_UTIL

The percentage of time that the CPU was idle during the interval. This is the total idle time, including waiting for I/O. On Unix systems, this is the same as the sum of the “%idle” and “%wio” fields reported by the “sar -u” command.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online.

On Solaris non-global zones, this metric is N/A. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_UTIL_CUM

The percentage of time that the CPU was idle over the cumulative collection time. This is the total idle time, including waiting for I/O.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_IDLE_UTIL_HIGH

The highest percentage of time that the CPU was idle during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_TIME

The time, in seconds, that the CPU spent processing interrupts during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On Hyper-V host, this metric is NA.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_TIME_CUM

The time, in seconds, that the CPU spent processing interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_UTIL

The percentage of time that the CPU spent processing interrupts during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On Hyper-V host, this metric is NA.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_UTIL_CUM

The percentage of time that the CPU spent processing interrupts over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_INTERRUPT_UTIL_HIGH

The highest percentage of time that the CPU spent processing interrupts during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_MT_ENABLED

On AIX, this metric indicates if this (Logical) System has SMT enabled or not.

Other platforms, this metric shows either HyperThreading(HT) is Enabled or Disabled/Not Supported.

On Linux, this state is dynamic: if HyperThreading is enabled but all the CPUs have only one logical processor enabled, this metric will report that HT is disabled.

On AIX System WPARs, this metric is NA.

On Windows, this metric will be "na" on Windows Server 2003 Itanium systems.

GBL_CPU_NICE_TIME

The time, in seconds, that the CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_TIME_CUM

The time, in seconds, that the CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_UTIL

The percentage of time that the CPU was in user mode at a nice priority during the interval.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_UTIL_CUM

The percentage of time that the CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NICE_UTIL_HIGH

The highest percentage of time during any one interval that the CPU was in user mode at a nice priority over the cumulative collection time.

On HP-UX, the NICE metrics include positive nice value CPU time only. Negative nice value CPU is broken out into NNICE (negative nice) metrics. Positive nice values range from 20 to 39. Negative nice values range from 0 to 19.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_NUM_THREADS

The number of active CPU threads supported by the CPU architecture.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be "na", some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

On AIX System WPARs, this metric is NA.

GBL_CPU_PHYSC

The number of physical processors utilized by the logical system.

On an Uncapped logical system (partition), this value will be equal to the physical processor capacity used by the logical system during the interval. This can be more than the value entitled for a logical system.

On a standalone system the value is calculated based on GBL_CPU_TOTAL_UTIL

GBL_CPU_PHYS_TOTAL_UTIL

The percentage of time the available physical CPUs were not idle for this logical system during the interval.

On AIX, this metric is calculated as :

$$\text{GBL_CPU_PHYS_TOTAL_UTIL} = \text{GBL_CPU_PHYS_USER_MODE_UTIL} + \text{GBL_CPU_PHYS_SYS_MODE_UTIL} ;$$
$$\text{GBL_CPU_PHYS_TOTAL_UTIL} + \text{GBL_CPU_PHYS_WAIT_UTIL} + \text{GBL_CPU_PHYS_IDLE_UTIL} = 100\%$$

On Power5 based systems, traditional sample based calculations cannot be made because the dispatch cycle for each of the virtual CPUs is not same. So Power5 processor maintains a per-thread register PURR. The thread is dispatching instructions or the thread that last dispatched an instruction will be incremented at every processor clock cycle. This makes the value to be distributed between the two threads. Power5 processor also maintains two more registers, one is timebase - which gets incremented at every tick and decremented - that provided periodic interrupts.

On a Shared LPAR environment, PURR is equal to the time that a virtual processor has spent on a physical processor. Hypervisor maintains a virtual timebase which is same as the sum of two PURRs.

On a Capped Shared logical system (partition), the calculations for the metric GBL_CPU_PHYS_USER_MODE_UTIL is as follows:

$$(\text{delta PURR in user mode/entitlement}) * 100$$

On an Uncapped Shared logical system (partition):
$$(\text{delta PURR in user mode/entitlement consumed}) * 100$$

The calculations for the other utilizations such as GBL_CPU_PHYS_USER_MODE_UTIL, GBL_CPU_PHYS_SYS_MODE_UTIL, and GBL_CPU_PHYS_WAIT_UTIL are also similar.

On a standalone system, the value will be equivalent to GBL_CPU_TOTAL_UTIL.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_SHARES_PRIO

The weightage/priority assigned to a Uncapped logical system. This value determines the minimum share of unutilized processing units that this logical system can utilize.

On AIX SPLPAR this value is dependent on the available processing units in the pool and can range from 0 to 255

On recognized VMware ESX guest, this value can range from 1 to 100000

On a standalone system the value will be "na".

GBL_CPU_SYS_MODE_TIME

The time, in seconds, that the CPU was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Hyper-V host, this metric indicates the time spent in Hypervisor code.

GBL_CPU_SYS_MODE_TIME_CUM

The time, in seconds, that the CPU was in system mode since over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "`-ignore_mt`" option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with "`-ignore_mt`" by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_SYS_MODE_UTIL

Percentage of time the CPU was in system mode during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

This metric is a subset of the `GBL_CPU_TOTAL_UTIL` percentage.

This is NOT a measure of the amount of time used by system daemon processes, since most system daemons spend part of their time in user mode and part in system calls, like any other process.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the "`-ignore_mt`" option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with "`-ignore_mt`" by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

High system mode CPU percentages are normal for IO intensive applications. Abnormally high system mode CPU percentages can indicate that a hardware problem is causing a high interrupt rate. It can also indicate programs that are not calling system calls efficiently. On a logical system, this metric indicates the percentage of time the logical processor was in kernel mode during this interval.

On Hyper-V host, this metric indicates the percentage of time spent in Hypervisor code.

GBL_CPU_SYS_MODE_UTIL_CUM

The percentage of time that the CPU was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_SYS_MODE_UTIL_HIGH

The highest percentage of time during any one interval that the CPU was in system mode over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TOTAL_TIME

The total time, in seconds, that the CPU was not idle in the interval.

This is calculated as

```
GBL_CPU_TOTAL_TIME =  
    GBL_CPU_USER_MODE_TIME +  
    GBL_CPU_SYS_MODE_TIME
```

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_TOTAL_TIME_CUM

The total time that the CPU was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_TOTAL_UTIL

Percentage of time the CPU was not idle during the interval.

This is calculated as

```
GBL_CPU_TOTAL_UTIL =
  GBL_CPU_USER_MODE_UTIL +
  GBL_CPU_SYS_MODE_UTIL
```

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

```
GBL_CPU_TOTAL_UTIL +
  GBL_CPU_IDLE_UTIL = 100%
```

This metric varies widely on most systems, depending on the workload. A consistently high CPU utilization can indicate a CPU bottleneck, especially when other indicators such as GBL_RUN_QUEUE and GBL_ACTIVE_PROC are also high. High CPU utilization can also occur on systems that are bottlenecked on memory, because the CPU spends more time paging and swapping.

NOTE: On Windows, this metric may not equal the sum of the APP_CPU_TOTAL_UTIL metrics. Microsoft states that “this is expected behavior” because this GBL_CPU_TOTAL_UTIL metric is taken from the performance library Processor objects while the APP_CPU_TOTAL_UTIL metrics are taken from the Process objects. Microsoft states that there can be CPU time accounted for in the Processor system objects that may not be seen in the Process objects. On a logical system, this metric indicates the logical utilization with respect to number of processors available for the logical system (GBL_NUM_CPU).

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TOTAL_UTIL_CUM

The percentage of total CPU time that the processor was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_TOTAL_UTIL_HIGH

The highest percentage of total CPU time during any one interval that the processor was not idle over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_USER_MODE_TIME

The time, in seconds, that the CPU was in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Hyper-V host, this metric indicates the time spent in guest code.

GBL_CPU_USER_MODE_TIME_CUM

The time, in seconds, that the CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On AIX System WPARs, this metric value is calculated against physical cpu time.

GBL_CPU_USER_MODE_UTIL

The percentage of time the CPU was in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

This metric is a subset of the GBL_CPU_TOTAL_UTIL percentage.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HPUX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

High user mode CPU percentages are normal for computation-intensive applications. Low values of user CPU utilization compared to relatively high values for GBL_CPU_SYS_MODE_UTIL can indicate an application or hardware problem. On a logical system, this metric indicates the percentage of time the logical processor was in user mode during this interval.

On Hyper-V host, this metric indicates the percentage of time spent in guest code.

GBL_CPU_USER_MODE_UTIL_CUM

The percentage of time that the CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_USER_MODE_UTIL_HIGH

The highest percentage of time during any one interval that the CPU was in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

GBL_CPU_WAIT_TIME

The time, in seconds, that the CPU was idle and there were processes waiting for physical IOs to complete during the interval.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On AIX System WPARs, this metric value is calculated against physical cpu time.

On Solaris non-global zones, this metric is N/A. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On Linux, this includes CPU steal time (shown as '%steal' in 'sar' and 'st' in 'vmstat').

GBL_CPU_WAIT_UTIL

The percentage of time during the interval that the CPU was idle and there were processes waiting for physical IOs to complete.

On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

On Solaris non-global zones, this metric is N/A. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

On Linux, this includes CPU steal time (shown as '%steal' in 'sar' and 'st' in 'vmstat').

GBL_CSWITCH_RATE

The average number of context switches per second during the interval.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

On Windows, this includes switches from one thread to another either inside a single process or across processes. A thread switch can be caused either by one thread asking another for information or by a thread being preempted by another higher priority thread becoming ready to run.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_CSWITCH_RATE_CUM

The average number of context switches per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

GBL_CSWITCH_RATE_HIGH

The highest number of context switches per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, this includes context switches that result in the execution of a different process and those caused by a process stopping, then resuming, with no other process running in the meantime.

GBL_DISK_PHYS_BYTE

The number of KBs transferred to and from disks during the interval. The bytes for all types of physical IOs are counted. Only local disks are counted in this measurement. NFS devices are excluded.

It is not directly related to the number of IOs, since IO requests can be of differing lengths.

On Unix systems, this includes file system IO, virtual memory IO, and raw IO.

On Windows, all types of physical IOs are counted.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_BYTE_RATE

The average number of KBs per second at which data was transferred to and from disks during the interval. The bytes for all types physical IOs are counted. Only local disks are counted in this measurement. NFS devices are excluded.

This is a measure of the physical data transfer rate. It is not directly related to the number of IOs, since IO requests can be of differing lengths.

This is an indicator of how much data is being transferred to and from disk devices. Large spikes in this metric can indicate a disk bottleneck.

On Unix systems, all types of physical disk IOs are counted, including file system, virtual memory, and raw reads.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_IO

The number of physical IOs during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk IOs are counted, including file system IO, virtual memory IO and raw IO.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_IO =
  GBL_DISK_FS_IO +
  GBL_DISK_VM_IO +
  GBL_DISK_SYSTEM_IO +
  GBL_DISK_RAW_IO
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_IO_CUM

The total number of physical IOs over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_IO_RATE

The number of physical IOs per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk IOs are counted, including file system IO, virtual memory IO and raw IO.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_IO_RATE =
  GBL_DISK_FS_IO_RATE +
  GBL_DISK_VM_IO_RATE +
  GBL_DISK_SYSTEM_IO_RATE +
  GBL_DISK_RAW_IO_RATE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_IO_RATE_CUM

The number of physical IOs per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ

The number of physical reads during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On HP-UX, there are many reasons why there is not a direct correlation between the number of logical IOs and physical IOs. For example, small sequential logical reads may be satisfied from the buffer cache, resulting in fewer physical IOs than logical IOs. Conversely, large logical IOs or small random IOs may result in more physical than logical IOs. Logical volume mappings, logical disk mirroring, and disk striping also tend to remove any correlation.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_READ =
  GBL_DISK_FS_READ +
  GBL_DISK_VM_READ +
  GBL_DISK_SYSTEM_READ +
  GBL_DISK_RAW_READ
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_BYTE

The number of KBs physically transferred from the disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_BYTE_CUM

The number of KBs (or MBs if specified) physically transferred from the disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_BYTE_RATE

The average number of KBs transferred from the disk per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_CUM

The total number of physical reads over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_PCT

The percentage of physical reads of total physical IO during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_PCT_CUM

The percentage of physical reads of total physical IO over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_READ_RATE

The number of physical reads per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk reads are counted, including file system, virtual memory, and raw reads.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_READ_RATE =
    GBL_DISK_FS_READ_RATE +
    GBL_DISK_VM_READ_RATE +
    GBL_DISK_SYSTEM_READ_RATE +
    GBL_DISK_RAW_READ_RATE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_READ_RATE_CUM

The average number of physical reads per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE

The number of physical writes during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On HP-UX, there are many reasons why there is not a direct correlation between logical IOs and physical IOs. For example, small logical writes may end up entirely in the buffer cache, and later generate fewer physical IOs when written to disk due to the larger IO size. Or conversely, small logical writes may require physical prefetching of the corresponding disk blocks before the data is merged and posted to disk. Logical volume mappings, logical disk mirroring, and disk striping also tend to remove any correlation.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_WRITE =
    GBL_DISK_FS_WRITE +
    GBL_DISK_VM_WRITE +
    GBL_DISK_SYSTEM_WRITE +
    GBL_DISK_RAW_WRITE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_WRITE_BYTE

The number of KBs (or MBs if specified) physically transferred to the disk during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_BYTE_CUM

The number of KBs (or MBs if specified) physically transferred to the disk over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_BYTE_RATE

The average number of KBs transferred to the disk per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_WRITE_CUM

The total number of physical writes over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_PCT

The percentage of physical writes of total physical IO during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

GBL_DISK_PHYS_WRITE_PCT_CUM

The percentage of physical writes of total physical IO over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_PHYS_WRITE_RATE

The number of physical writes per second during the interval. Only local disks are counted in this measurement. NFS devices are excluded.

On Unix systems, all types of physical disk writes are counted, including file system IO, virtual memory IO, and raw writes.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On HP-UX, this is calculated as

```
GBL_DISK_PHYS_WRITE_RATE =  
    GBL_DISK_FS_WRITE_RATE +  
    GBL_DISK_VM_WRITE_RATE +  
    GBL_DISK_SYSTEM_WRITE_RATE +  
    GBL_DISK_RAW_WRITE_RATE
```

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_PHYS_WRITE_RATE_CUM

The number of physical writes per second over the cumulative collection time. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

GBL_DISK_REQUEST_QUEUE

The total length of all of the disk queues at the end of the interval.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

On SUN, if a CD drive is powered off, or no CD is inserted in the CD drive at boottime, the operating system does not provide performance data for that device. This can be determined by checking the "by-disk" data when provided in a product. If the CD drive has an entry in the list of active disks on a system, then data for that device is being collected.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_TIME_PEAK

The time, in seconds, during the interval that the busiest disk was performing IO transfers. This is for the busiest disk only, not all disk devices. This counter is based on an end-to-end measurement for each IO transfer updated at queue entry and exit points.

Only local disks are counted in this measurement. NFS devices are excluded.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_UTIL

On HP-UX, this is the average percentage of time during the interval that all disks had IO in progress from the point of view of the Operating System. This is the average utilization for all disks.

On all other Unix systems, this is the average percentage of disk in use time of the total interval (that is, the average utilization).

Only local disks are counted in this measurement. NFS devices are excluded.

GBL_DISK_UTIL_PEAK

The utilization of the busiest disk during the interval.

On HP-UX, this is the percentage of time during the interval that the busiest disk device had IO in progress from the point of view of the Operating System.

On all other systems, this is the percentage of time during the interval that the busiest disk was performing IO transfers.

It is not an average utilization over all the disk devices. Only local disks are counted in this measurement. NFS devices are excluded.

Some Linux kernels, typically 2.2 and older kernels, do not support the instrumentation needed to provide values for this metric. This metric will be "na" on the affected kernels. The "sar -d" command will also not be present on these systems. Distributions and OS releases that are known to be affected include: TurboLinux 7, SuSE 7.2, and Debian 3.0.

A peak disk utilization of more than 50 percent often indicates a disk IO subsystem bottleneck situation. A bottleneck may not be in the physical disk drive itself, but elsewhere in the IO path.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_DISK_UTIL_PEAK_CUM

The average utilization of the busiest disk in each interval over the cumulative collection time. Utilization is the percentage of time in use versus the time in the measurement interval. For each interval a different disk may be the busiest. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_DISK_UTIL_PEAK_HIGH

The highest utilization of any disk during any interval over the cumulative collection time. Utilization is the percentage of time in use versus the time in the measurement interval. Only local disks are counted in this measurement. NFS devices are excluded.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_DISTRIBUTION

The software distribution, if available.

GBL_FS_SPACE_UTIL_PEAK

The percentage of occupied disk space to total disk space for the fullest file system found during the interval. Only locally mounted file systems are counted in this metric.

This metric can be used as an indicator that at least one file system on the system is running out of disk space.

On Unix systems, CDROM and PC file systems are also excluded. This metric can exceed 100 percent. This is because a portion of the file system space is reserved as a buffer and can only be used by root. If the root user has made the file system grow beyond the reserved buffer, the utilization will be greater than 100 percent. This is a dangerous situation since if the root user totally fills the file system, the system may crash.

On Windows, CDROM file systems are also excluded.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_GMTOFFSET

The difference, in minutes, between local time and GMT (Greenwich Mean Time).

GBL_IGNORE_MT

This boolean value indicates whether the CPU normalization is on or off. If the metric value is "true", CPU related metrics in the global class will report values which are normalized against the number of active cores on the system.

If the metric value is "false", CPU related metrics in the global class will report values which are normalized against the number of CPU threads on the system.

If CPU MultiThreading is turned off this configuration option is a no-op and the metric value will be "true".

On Linux, this metric will only report "true" if this configuration is on and if the kernel provides enough information to determine whether MultiThreading is turned on.

On HP-UX, this metric will report "na" if the processor doesn't support the feature.

GBL_INTERRUPT

The number of IO interrupts during the interval.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_INTERRUPT_RATE

The average number of IO interrupts per second during the interval.

On HP-UX and SUN this value includes clock interrupts. To get non-clock device interrupts, subtract clock interrupts from the value.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

GBL_INTERRUPT_RATE_CUM

The average number of IO interrupts per second over the cumulative collection time.

On HP-UX and SUN this value includes clock interrupts. To get non-clock device interrupts, subtract clock interrupts from the value.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_INTERRUPT_RATE_HIGH

The highest number of IO interrupts per second during any one interval over the cumulative collection time.

On HP-UX and SUN this value includes clock interrupts. To get non-clock device interrupts, subtract clock interrupts from the value.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_INTERVAL

The amount of time in the interval.

This measured interval is slightly larger than the desired or configured interval if the collection program is delayed by a higher priority process and cannot sample the data immediately.

GBL_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_JAVAARG

This boolean value indicates whether the java class overloading mechanism is enabled or not. This metric will be set when the javaarg flag in the parm file is set. The metric affected by this setting is PROC_PROC_ARGV1. This setting is useful to construct parm file java application definitions using the argv1= keyword.

GBL_LOADAVG

The 1 minute load average of the system obtained at the time of logging.

On windows this is the load average of the system over the interval. Load average on windows is the average number of threads that have been waiting in ready state during the interval. This is obtained by checking the number of threads in ready state every sub proc interval, accumulating them over the interval and averaging over the interval.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_LOADAVG15

The 15 minute load average of the system obtained at the time of logging.

GBL_LOADAVG5

The 5 minute load average of the system obtained at the time of logging.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_LOADAVG_CUM

The average load average of the system over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_LOADAVG_HIGH

The highest value of the load average during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_LOST_MI_TRACE_BUFFERS

The number of trace buffers lost by the measurement processing daemon.

On HP-UX systems, if this value is > 0, the measurement subsystem is not keeping up with the system events that generate traces.

For other Unix systems, if this value is > 0, the measurement subsystem is not keeping up with the ARM API calls that generate traces.

Note: The value reported for this metric will roll over to 0 once it crosses INTMAX.

GBL_LS_MODE

Indicates whether the CPU entitlement for the logical system is Capped or Uncapped.

On a recognized VMware ESX guest, where VMware guest SDK is enabled, the value is "Uncapped" if maximum CPU entitlement (GBL_CPU_ENTL_MAX) is unlimited.

Else, the value is always "Capped".

GBL_LS_ROLE

Indicates whether Perf Agent is installed on Logical system or host or standalone system. This metric will be either "GUEST", "HOST" or "STAND".

GBL_LS_SHARED

In a virtual environment, this metric indicates whether the physical CPUs are dedicated to this Logical system or shared.

On AIX SPLPAR, this metric is equivalent to "Type" field of 'lparstat -i' command.

On a recognized VMware ESX guest, where VMware guest SDK is enabled, the value is "Shared".

On a standalone system the value of this metrics is "Dedicated".

On AIX System WPARs, this metric is NA.

GBL_LS_TYPE

The virtualization technology if applicable. The value of this metric is "HPVM" on HP-UX host, "LPAR" on AIX LPAR, "Sys WPAR" on system WPAR, "Zone" on Solaris Zones, "VMware" on recognized VMware ESX guest and VMware ESX Server console, "Hyper-V" on Hyper-V host, else "NoVM".

In conjunction with GBL_LS_ROLE this metric could be used to identify the environment in which Perf Agent/Glance is running. For example, if GBL_LS_ROLE is "Guest" and GBL_LS_TYPE is "VMware" then PA/Glance is running on a VMware Guest.

GBL_MACHINE

An ASCII string representing the Processor Architecture. And machine hardware model is represented by GBL_MACHINE_MODEL metric.

GBL_MACHINE_MEM_USED

The amount of physical host memory currently consumed for this logical system's physical memory. On a standalone system, the value will be $(\text{GBL_MEM_UTIL} * \text{GBL_MEM_PHYS}) / 100$

GBL_MACHINE_MODEL

The CPU model. This is similar to the information returned by the GBL_MACHINE metric and the uname command(except for Solaris 10 x86/x86_64). However, this metric returns more information on some processors.

On HP-UX, this is the same information returned by the model command.

GBL_MEM_AVAIL

The amount of physical available memory in the system (in MBs unless otherwise specified).

On Windows, memory resident operating system code and data is not included as available memory.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_CACHE

The amount of physical memory (in MBs unless otherwise specified) used by the buffer cache during the interval.

On HP-UX 11i v2 and below, the buffer cache is a memory pool used by the system to stage disk IO data for the driver.

On HP-UX 11i v3 and above this metric value represents the usage of the file system buffer cache which is still being used for file system metadata.

On SUN, this value is obtained by multiplying the system page size times the number of buffer headers (nbuf). For example, on a SPARCstation 10 the buffer size is usually $(200 \text{ (page size buffers)} * 4096 \text{ (bytes/page)}) = 800 \text{ KB}$.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The "nbuf" value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On AIX, this value should be minimal since most disk IOs are done through memory mapped files.

GBL_MEM_CACHE_UTIL

The percentage of physical memory used by the buffer cache during the interval.

On HP-UX 11i v2 and below, the buffer cache is a memory pool used by the system to stage disk IO data for the driver.

On HP-UX 11i v3 and above this metric value represents the usage of the file system buffer cache which is still being used for file system metadata.

On SUN, this percentage is based on calculating the buffer cache size by multiplying the system page size times the number of buffer headers (nbuf). For example, on a SPARCstation 10 the buffer size is usually $(200 \text{ (page size buffers)} * 4096 \text{ (bytes/page)}) = 800 \text{ KB}$.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The "nbuf" value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On AIX, this value should be minimal since most disk IOs are done through memory mapped files. On Windows the value reports 'copy read hit %' and 'Pin read hit %'.

GBL_MEM_ENTL_MAX

In a virtual environment, this metric indicates the maximum amount of memory configured for this logical system. The value is -3 if entitlement is 'Unlimited' for this logical system.

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na"

On Solaris non-global zones, this metric value is equivalent to 'capped-memory' value for 'zonecfg -z zonename info' command.

On a standalone system this metric is equivalent to GBL_MEM_PHYS.

GBL_MEM_ENTL_MIN

In a virtual environment, this metric indicates the minimum amount of memory configured for this logical system.

On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na"

On a standalone system, this metrics is equivalent to GBL_MEM_PHYS.

GBL_MEM_FILE_PAGEIN_RATE

The number of page ins from the file system per second during the interval.

On Solaris, this is the same as the "fpi" value from the "vmstat -p" command, divided by page size in KB.

On Linux, the value is reported in kilobytes and matches the 'io/bi' values from vmstat.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_FILE_PAGEOUT_RATE

The number of page outs to the file system per second during the interval.

On Solaris, this is the same as the "fpo" value from the "vmstat -p" command, divided by page size in KB.

On Linux, the value is reported in kilobytes and matches the 'io/bo' values from vmstat.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_FILE_PAGE_CACHE

The amount of physical memory (in MBs unless otherwise specified) used by the file cache during the interval. File cache is a memory pool used by the system to stage disk IO data for the driver.

This metric is supported on HP-UX 11iv3 and above. The `filecache_min` and `filecache_max` tunables control the filecache memory usage on the system. The `filecache_min` tunable specifies the amount of physical memory that is guaranteed to be available for filecache on the system. The filecache memory usage can grow beyond `filecache_min`, up to the limit set by the `filecache_max` tunable. The Virtual Memory (VM) subsystem always pre reserves 'filecache_min' tunable value worth of pages on the system for filecache, even in the case of filecache under utilization (actual filecache utilization < `filecache_min` value). This preserved memory by the VM is not available for the user. In this scenario, this metric will show the 'filecache_min' as the filecache value, rather than showing the actual filecache utilization.

On Linux, this metric is equal to 'cached' value of 'free -m' command output.

GBL_MEM_FILE_PAGE_CACHE_UTIL

The percentage of physical memory used by the file cache during the interval. File cache is a memory pool used by the system to stage disk IO data for the driver.

This metric is supported on HP-UX 11iv3 and above. The `filecache_min` and `filecache_max` tunables control the filecache memory usage on the system. The `filecache_min` tunable specifies the amount of physical memory that is guaranteed to be available for filecache on the system. The filecache memory usage can grow beyond `filecache_min`, up to the limit set by the `filecache_max` tunable. The Virtual Memory (VM) subsystem always pre reserves 'filecache_min' tunable value worth of pages on the system for filecache, even in the case of filecache under utilization (actual filecache utilization < `filecache_min` value). This preserved memory by the VM is not available for the user. In this scenario, this metric will show the 'filecache_min' as the filecache value, rather than showing the actual filecache utilization.

On Linux, this metric is derived from 'cached' value of 'free -m' command output.

GBL_MEM_FREE

The amount of memory not allocated (in MBs unless otherwise specified). As this value drops, the likelihood increases that swapping or paging out to disk may occur to satisfy new memory requests.

On SUN, low values for this metric may not indicate a true memory shortage. This metric can be influenced by the VMM (Virtual Memory Management) system. On Linux, this metric is sum of 'free' and 'cached' memory.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

Locality Domain metrics are available on HP-UX 11iv2 and above. `GBL_MEM_FREE` and `LDOM_MEM_FREE`, as well as the memory utilization metrics derived from them, may not always fully match. `GBL_MEM_FREE` represents free memory in the kernel's reservation layer while `LDOM_MEM_FREE` shows actual free pages. If memory has been reserved but not actually consumed from the Locality Domains, the two values won't match. Because `GBL_MEM_FREE` includes pre-reserved memory, the `GBL_MEM_*` metrics are a better indicator of actual memory consumption in most situations.

GBL_MEM_FREE_UTIL

The percentage of physical memory that was free at the end of the interval.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_OVERHEAD

The amount of "overhead" memory associated with this logical system that is currently consumed on the host system. On VMware ESX Server console, the value is equivalent to sum of the current overhead memory for all running virtual machines. On a standalone system, the value will be 0. On a recognized VMware ESX guest, where VMware guest SDK is disabled, the value is "na".

GBL_MEM_PAGEIN

The total number of page ins from the disk during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX, this is the same as the "page ins" value from the "vmstat -s" command. On AIX, this is the same as the "paging space page ins" value. Remember that "vmstat -s" reports cumulative counts.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEIN_BYTE

The number of KBs (or MBs if specified) of page ins during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_CUM

The number of KBs (or MBs if specified) of page ins over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_RATE

The number of KBs per second of page ins during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_RATE_CUM

The average number of KBs per second of page ins over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_BYTE_RATE_HIGH

The highest number of KBs per second of page ins during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_CUM

The total number of page ins from the disk over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_RATE

The total number of page ins per second from the disk during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX and AIX, this is the same as the "pi" value from the vmstat command.

On Solaris, this is the same as the sum of the "epi" and "api" values from the "vmstat -p" command, divided by the page size in KB.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEIN_RATE_CUM

The average number of page ins per second over the cumulative collection time. This includes pages paged in from paging space and, except for AIX, from the file system.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEIN_RATE_HIGH

The highest number of page ins per second from disk during any interval over the cumulative collection time.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT

The total number of page outs to the disk during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX, this is the same as the "page outs" value from the "vmstat -s" command. On HP-UX 11iv3 and above this includes filecache page outs also. On AIX, this is the same as the "paging space page outs" value. Remember that "vmstat -s" reports cumulative counts.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_BYTE

The number of KBs (or MBs if specified) of page outs during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_BYTE_CUM

The number of KBs (or MBs if specified) of page outs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_BYTE_RATE

The number of KBs (or MBs if specified) per second of page outs during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_BYTE_RATE_CUM

The average number of KBs per second of page outs over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_BYTE_RATE_HIGH

The highest number of KBs per second of page outs during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_CUM

The total number of page outs to the disk over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_RATE

The total number of page outs to the disk per second during the interval.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

On HP-UX and AIX, this is the same as the "po" value from the vmstat command.

On Solaris, this is the same as the sum of the "epo" and "apo" values from the "vmstat -p" command, divided by the page size in KB.

On Windows, this counter also includes paging traffic on behalf of the system cache to access file data for applications and so may be high when there is no memory pressure.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGEOUT_RATE_CUM

The average number of page outs to the disk per second over the cumulative collection time. This includes pages paged out to paging space and, except for AIX, to the file system.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGEOUT_RATE_HIGH

The highest number of page outs per second to disk during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, Linux and AIX, this reflects paging activity between memory and paging space. It does not include activity between memory and file systems.

On Windows, this includes paging activity for both file systems and paging space.

GBL_MEM_PAGE_FAULT

The number of page faults that occurred during the interval.

On Linux this metric is available only on 2.6 and above kernel versions.

GBL_MEM_PAGE_FAULT_CUM

The number of page faults that occurred over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_MEM_PAGE_FAULT_RATE

The number of page faults per second during the interval.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGE_FAULT_RATE_CUM

The average number of page faults per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_MEM_PAGE_FAULT_RATE_HIGH

The highest page fault per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_MEM_PAGE_REQUEST

The number of page requests to or from the disk during the interval.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

On HP-UX, this is the same as the sum of the "page ins" and "page outs" values from the "vmstat -s" command. On AIX, this is the same as the sum of the "paging space page ins" and "paging space page outs" values. Remember that "vmstat -s" reports cumulative counts.

On Windows, this counter also includes paging traffic on behalf of the system cache to access file data for applications and so may be high when there is no memory pressure.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGE_REQUEST_CUM

The total number of page requests to or from the disk over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

On Windows, this counter also includes paging traffic on behalf of the system cache to access file data for applications and so may be high when there is no memory pressure.

GBL_MEM_PAGE_REQUEST_RATE

The number of page requests to or from the disk per second during the interval.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

On HP-UX and AIX, this is the same as the sum of the "pi" and "po" values from the vmstat command.

On Solaris, this is the same as the sum of the "epi", "epo", "api", and "apo" values from the "vmstat -p" command, divided by the page size in KB.

Higher than normal rates can indicate either a memory or a disk bottleneck. Compare GBL_DISK_UTIL_PEAK and GBL_MEM_UTIL to determine which resource is more constrained. High rates may also indicate memory thrashing caused by a particular application or set of applications. Look for processes with high major fault rates to identify the culprits.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PAGE_REQUEST_RATE_CUM

The average number of page requests to or from the disk per second over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

GBL_MEM_PAGE_REQUEST_RATE_HIGH

The highest number of page requests per second during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, Solaris, and AIX, this includes pages paged to or from the paging space and not to or from the file system.

On Windows, this includes pages paged to or from both paging space and the file system.

GBL_MEM_PHYS

The amount of physical memory in the system (in MBs unless otherwise specified).

On HP-UX, banks with bad memory are not counted. Note that on some machines, the Processor Dependent Code (PDC) code uses the upper 1MB of memory and thus reports less than the actual physical memory of the system. Thus, on a system with 256MB of physical memory, this metric and dmesg(1M) might only report 267,386,880 bytes (255MB). This is all the physical memory that software on the machine can access.

On Windows, this is the total memory available, which may be slightly less than the total amount of physical memory present in the system. This value is also reported in the Control Panel's About Windows NT help topic.

On Linux, this is the amount of memory given by dmesg(1M). If the value is not available in kernel ring buffer, then the sum of system memory and available memory will be reported as physical memory.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_PHYS_SWAPPED

On a recognized VMware ESX guest, where VMware guest SDK is enabled, this metrics indicates the amount of memory that has been reclaimed by ESX Server from this logical system by transparently swapping logical system's memory to disk. The value is "na" otherwise.

GBL_MEM_SHARES_PRIO

The weightage/priority for memory assigned to this logical system. This value influences the share of unutilized physical Memory that this logical system can utilize. On a recognized VMware ESX guest, where VMware guest SDK is enabled, this value can range from 0 to 100000. The value will be "na" otherwise.

GBL_MEM_SWAPIN_BYTE

The number of KBs transferred in from disk due to swap ins (or reactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPIN_BYTE_CUM

The number of KBs transferred in from disk due to swap ins (or reactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_BYTE_RATE

The number of KBs per second transferred from disk due to swap ins (or reactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller

than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPIN_BYTE_RATE_CUM

The number of KBs per second transferred from disk due to swap ins (or reactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPIN_BYTE_RATE_HIGH

The highest number of KBs per second transferred from disk due to swap ins (or reactivations on HP-UX) during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_BYTE

The number of KBs (or MBs if specified) transferred out to disk due to swap outs (or deactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise,

swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPOUT_BYTE_CUM

The number of KBs (or MBs if specified) transferred out to disk due to swap outs (or deactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_BYTE_RATE

The number of KBs (or MBs if specified) per second transferred out to disk due to swap outs (or deactivations on HP-UX) during the interval.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

On Solaris non-global zones with Uncapped Memory scenario, this metric value is same as seen in global zone.

GBL_MEM_SWAPOUT_BYTE_RATE_CUM

The average number of KBs (or MBs if specified) per second transferred out to disk due to swap outs (or deactivations on HP-UX) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise,

swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SWAPOUT_BYTE_RATE_HIGH

The highest number of KBs (or MBs if specified) per second transferred out to disk due to swap outs (or deactivations on HP-UX) during any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Linux and AIX, swap metrics are equal to the corresponding page metrics.

On HP-UX, process swapping was replaced by a combination of paging and deactivation. Process deactivation occurs when the system is thrashing or when the amount of free memory falls below a critical level. The swapper then marks certain processes for deactivation and removes them from the run queue. Pages within the associated memory regions are reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.

To summarize, a process swap-out on HP-UX is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than HP-UX 9.x counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.

GBL_MEM_SYS

The amount of physical memory (in MBs unless otherwise specified) used by the system (kernel) during the interval. System memory does not include the buffer cache. On HP-UX and Linux this does not include filecache also.

On HP-UX 11.0, this metric does not include some kinds of dynamically allocated kernel memory. This has always been reported in the GBL_MEM_USER* metrics.

On HP-UX 11.11 and beyond, this metric includes some kinds of dynamically allocated kernel memory.

On Solaris non-global zones, this metric shows value as 0.

GBL_MEM_SYS_UTIL

The percentage of physical memory used by the system during the interval.

System memory does not include the buffer cache. On HP-UX and Linux this does not include filecache also.

On HP-UX 11.0, this metric does not include some kinds of dynamically allocated kernel memory. This has always been reported in the GBL_MEM_USER* metrics.

On HP-UX 11.11 and beyond, this metric includes some kinds of dynamically allocated kernel memory.

On Solaris non-global zones, this metric shows value as 0.

GBL_MEM_USER

The amount of physical memory (in MBs unless otherwise specified) allocated to user code and data at the end of the interval. User memory regions include code, heap, stack, and other data areas including shared memory. This does not include memory for buffer cache. On HP-UX and Linux this does not include filecache also.

On HP-UX 11.0, this metric includes some kinds of dynamically allocated kernel memory.

On HP-UX 11.11 and beyond, this metric does not include some kinds of dynamically allocated kernel memory. This is now reported in the GBL_MEM_SYS* metrics.

Large fluctuations in this metric can be caused by programs which allocate large amounts of memory and then either release the memory or terminate. A slow continual increase in this metric may indicate a program with a memory leak.

GBL_MEM_USER_UTIL

The percent of physical memory allocated to user code and data at the end of the interval. This metric shows the percent of memory owned by user memory regions such as user code, heap, stack and other data areas including shared memory. This does not include memory for buffer cache. On HP-UX and Linux this does not include filecache also. On HP-UX 11.0, this metric includes some kinds of dynamically allocated kernel memory.

On HP-UX 11.11 and beyond, this metric does not include some kinds of dynamically allocated kernel memory. This is now reported in the GBL_MEM_SYS* metrics.

Large fluctuations in this metric can be caused by programs which allocate large amounts of memory and then either release the memory or terminate. A slow continual increase in this metric may indicate a program with a memory leak.

GBL_MEM_UTIL

The percentage of physical memory in use during the interval. This includes system memory (occupied by the kernel), buffer cache and user memory.

On HP-UX 11iv3 and above, this includes file cache also.

On HP-UX, this calculation is done using the byte values for physical memory and used memory, and is therefore more accurate than comparing the reported kilobyte values for physical memory and used memory.

On SUN, high values for this metric may not indicate a true memory shortage. This metric can be influenced by the VMM (Virtual Memory Management) system.

Locality Domain metrics are available on HP-UX 11iv2 and above. GBL_MEM_FREE and LDOM_MEM_FREE, as well as the memory utilization metrics derived from them, may not always fully match. GBL_MEM_FREE represents free memory in the kernel's reservation layer while LDOM_MEM_FREE shows actual free pages. If memory has been reserved but not actually consumed from the Locality Domains, the two values won't match. Because GBL_MEM_FREE includes pre-reserved memory, the GBL_MEM_* metrics are a better indicator of actual memory consumption in most situations.

GBL_MEM_UTIL_CUM

The average percentage of physical memory in use over the cumulative collection time. This includes system memory (occupied by the kernel), buffer cache and user memory.

On HP-UX 11iv3 and above, this includes file cache also.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_MEM_UTIL_HIGH

The highest percentage of physical memory in use in any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_NET_COLLISION

The number of collisions that occurred on all network interfaces during the interval. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not include deferred packets.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the "Single Collision Frames", "Multiple Collision Frames", "Late Collisions", and "Excessive Collisions" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Coll" column from the "netstat -i" command ("collisions" from the "netstat -i -e" command on Linux) for a network device. See also netstat(1).

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_COLLISION_1_MIN_RATE

The number of collisions per minute on all network interfaces during the interval. This metric does not include deferred packets.

This does not include data for loopback interface.

Collisions occur on any busy network, but abnormal collision rates could indicate a hardware or software problem.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_COLLISION_CUM

The number of collisions that occurred on all network interfaces over the cumulative collection time. A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested. This metric does not include deferred packets.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For HP-UX, this will be the same as the sum of the "Single Collision Frames", "Multiple Collision Frames", "Late Collisions", and "Excessive Collisions" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. For this release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For other Unix systems, this is the same as the sum of the "Coll" column from the "netstat -i" command ("collisions" from the "netstat -i -e" command on Linux) for a network device. See also netstat(1).

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_COLLISION_PCT

The percentage of collisions to total outbound packet attempts during the interval. Outbound packet attempts include both successful packets and collisions.

This does not include data for loopback interface.

A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.

This metric does not currently include deferred packets.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_COLLISION_PCT_CUM

The percentage of collisions to total outbound packet attempts over the cumulative collection time. Outbound packet attempts include both successful packets and collisions.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.

This metric does not currently include deferred packets.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_COLLISION_RATE

The number of collisions per second on all network interfaces during the interval. This metric does not include deferred packets.

This does not include data for loopback interface.

A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.

AIX does not support the collision count for the ethernet interface. The collision count is supported for the token ring (tr) and loopback (lo) interfaces. For more information, please refer to the netstat(1) man page.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_ERROR

The number of errors that occurred on all network interfaces during the interval.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the "Inbound Errors" and "Outbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "lerrs" (RX-ERR on Linux) and "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR_1_MIN_RATE

The number of errors per minute on all network interfaces during the interval. This rate should normally be zero or very small. A large error rate can indicate a hardware or software problem.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR_CUM

The number of errors that occurred on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For HP-UX, this will be the same as the total sum of the "Inbound Errors" and "Outbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "lerrs" (RX-ERR on Linux) and "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_ERROR_RATE

The number of errors per second on all network interfaces during the interval.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_ERROR

The number of inbound errors that occurred on all network interfaces during the interval.

A large number of errors may indicate a hardware problem on the network.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the "Inbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "Ierrs" (RX-ERR on Linux) and "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_ERROR_CUM

The number of inbound errors that occurred on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

A large number of errors may indicate a hardware problem on the network.

For HP-UX, this will be the same as the total sum of the "Inbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "Ierrs" (RX-ERR on Linux) and "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_ERROR_PCT

The percentage of inbound network errors to total inbound packet attempts during the interval. Inbound packet attempts include both packets successfully received and those that encountered errors.

This does not include data for loopback interface.

A large number of errors may indicate a hardware problem on the network. The percentage of inbound errors to total packets attempted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_ERROR_PCT_CUM

The percentage of inbound network errors to total inbound packet attempts over the cumulative collection time. Inbound packet attempts include both packets successfully received and those that encountered errors.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

A large number of errors may indicate a hardware problem on the network. The percentage of inbound errors to total packets attempted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_ERROR_RATE

The number of inbound errors per second on all network interfaces during the interval.

This does not include data for loopback interface.

A large number of errors may indicate a hardware problem on the network. The percentage of inbound errors to total packets attempted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_ERROR_RATE_CUM

The average number of inbound errors per second on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_PACKET

The number of successful packets received through all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the "Inbound Unicast Packets" and "Inbound Non-Unicast Packets" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Ipkts" column (RX-OK on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_IN_PACKET_CUM

The number of successful packets received through all network interfaces over the cumulative collection time.

Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For HP-UX, this will be the same as the total sum of the "Inbound Unicast Packets" and "Inbound Non-Unicast Packets" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Ipkts" column (RX-OK on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_IN_PACKET_RATE

The number of successful packets per second received through all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_ERROR

The number of outbound errors that occurred on all network interfaces during the interval.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the "Outbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR_CUM

The number of outbound errors that occurred on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For HP-UX, this will be the same as the total sum of the "Outbound Errors" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of "Oerrs" (TX-ERR on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR_PCT

The percentage of outbound network errors to total outbound packet attempts during the interval. Outbound packet attempts include both packets successfully sent and those that encountered errors.

This does not include data for loopback interface.

The percentage of outbound errors to total packets attempted to be transmitted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_ERROR_PCT_CUM

The percentage of outbound network errors to total outbound packet attempts over the cumulative collection time. Outbound packet attempts include both packets successfully sent and those that encountered errors.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

The percentage of outbound errors to total packets attempted to be transmitted should remain low.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_ERROR_RATE

The number of outbound errors per second on all network interfaces during the interval.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_ERROR_RATE_CUM

The number of outbound errors per second on all network interfaces over the cumulative collection time.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_PACKET

The number of successful packets sent through all network interfaces during the last interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

For HP-UX, this will be the same as the sum of the "Outbound Unicast Packets" and "Outbound Non-Unicast Packets" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Opkts" column (TX-OK on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_OUT_PACKET_CUM

The number of successful packets sent through all network interfaces over the cumulative collection time. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

For HP-UX, this will be the same as the total sum of the "Outbound Unicast Packets" and "Outbound Non-Unicast Packets" values from the output of the "lanadmin" utility for the network interface. Remember that "lanadmin" reports cumulative counts. As of the HP-UX 11.0 release and beyond, "netstat -i" shows network activity on the logical level (IP) only.

For all other Unix systems, this is the same as the sum of the "Opkts" column (TX-OK on Linux) from the "netstat -i" command for a network device. See also netstat(1).

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

GBL_NET_OUT_PACKET_RATE

The number of successful packets per second sent through the network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NET_PACKET

The total number of successful inbound and outbound packets for all network interfaces during the interval. These are the packets that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

GBL_NET_PACKET_RATE

The number of successful packets per second (both inbound and outbound) for all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.

This does not include data for loopback interface.

This metric is updated at the sampling interval, regardless of the number of IP addresses on the system.

On Windows system, the packet size for NBT connections is defined as 1 Kbyte.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_NFS_CALL

The number of NFS calls the local system has made as either a NFS client or server during the interval.

This includes both successful and unsuccessful calls. Unsuccessful calls are those that cannot be completed due to resource limitations or LAN packet errors.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

On AIX System WPARs, this metric is NA.

GBL_NFS_CALL_RATE

The number of NFS calls per second the system made as either a NFS client or NFS server during the interval.

Each computer can operate as both a NFS server, and as an NFS client.

This metric includes both successful and unsuccessful calls. Unsuccessful calls are those that cannot be completed due to resource limitations or LAN packet errors.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

On AIX System WPARs, this metric is NA.

GBL_NFS_CLIENT_BAD_CALL

The number of failed NFS client calls during the interval. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

GBL_NFS_CLIENT_BAD_CALL_CUM

The number of failed NFS client calls over the cumulative collection time. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_NFS_CLIENT_CALL

The number of NFS calls the local machine has processed as a NFS client during the interval. Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate should exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

GBL_NFS_CLIENT_CALL_CUM

The number of NFS calls the local machine has processed as a NFS client over the cumulative collection time. Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate should exceed the IO rate. This metric includes both successful and unsuccessful calls.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

GBL_NFS_CLIENT_CALL_RATE

The number of NFS calls the local machine has processed as a NFS client per second during the interval. Calls are the system call used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate should exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

GBL_NFS_CLIENT_IO

The number of NFS IOs the local machine has completed as an NFS client during the interval. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_CLIENT_IO_CUM

The number of NFS IOs the local machine has completed as an NFS client over the cumulative collection time. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_CLIENT_IO_PCT

The percentage of NFS IOs the local machine has completed as an NFS client versus total NFS IOs completed during the interval. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

A percentage greater than 50 indicates that this machine is acting more as a client. A percentage less than 50 indicates this machine is acting more as a server for others.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_CLIENT_IO_PCT_CUM

The percentage of NFS IOs the local machine has completed as an NFS client versus total NFS IOs completed over the cumulative collection time. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

A percentage greater than 50 indicates that this machine is acting more as a client. A percentage less than 50 indicates this machine is acting more as a server for others.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_CLIENT_IO_RATE

The number of NFS IOs per second the local machine has completed as an NFS client during the interval. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_CLIENT_IO_RATE_CUM

The number of NFS IOs per second the local machine has completed as an NFS client over the cumulative collection time. This number represents physical IOs sent by the client in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both an NFS server, and as a NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS IOs include reads and writes from successful calls to `getattr`, `setattr`, `lookup`, `read`, `readdir`, `readlink`, `write`, and `writocache`.

GBL_NFS_CLIENT_READ_RATE

The number of NFS “read” operations per second the system generated as an NFS client during the interval.

NFS Version 2 read operations consist of `getattr`, `lookup`, `readlink`, `readdir`, `null`, `root`, `statfs`, and `read`.

NFS Version 3 read operations consist of `getattr`, `lookup`, `access`, `readlink`, `read`, `readdir`, `readdirplus`, `fsstat`, `fsinfo`, and `null`.

GBL_NFS_CLIENT_READ_RATE_CUM

The average number of NFS “read” operations per second the system generated as an NFS client over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS Version 2 read operations consist of `getattr`, `lookup`, `readlink`, `readdir`, `null`, `root`, `statfs`, and `read`.

NFS Version 3 read operations consist of `getattr`, `lookup`, `access`, `readlink`, `read`, `readdir`, `readdirplus`, `fsstat`, `fsinfo`, and `null`.

GBL_NFS_CLIENT_WRITE_RATE

The number of NFS “write” operations per second the system generated as an NFS client during the interval.

NFS Version 2 write operations consist of `setattr`, `write`, `writocache`, `create`, `remove`, `rename`, `link`, `symlink`, `mkdir`, and `rmdir`.

NFS Version 3 write operations consist of `setattr`, `write`, `create`, `mkdir`, `symlink`, `mknod`, `remove`, `rmdir`, `rename`, `link`, `pathconf`, and `commit`.

GBL_NFS_CLIENT_WRITE_RATE_CUM

The average number of NFS “write” operations per second the system generated as an NFS client over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS Version 2 write operations consist of `setattr`, `write`, `writocache`, `create`, `remove`, `rename`, `link`, `symlink`, `mkdir`, and `rmdir`.

NFS Version 3 write operations consist of `setattr`, `write`, `create`, `mkdir`, `symlink`, `mknod`, `remove`, `rmdir`, `rename`, `link`, `pathconf`, and `commit`.

GBL_NFS_SERVER_BAD_CALL

The number of failed NFS server calls during the interval. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

GBL_NFS_SERVER_BAD_CALL_CUM

The number of failed NFS server calls over the cumulative collection time. Calls fail due to lack of system resources (lack of virtual memory) as well as network errors.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_NFS_SERVER_CALL

The number of NFS calls the local machine has processed as a NFS server during the interval.

Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate could exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

GBL_NFS_SERVER_CALL_CUM

The number of NFS calls the local machine has processed as a NFS server over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate could exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

GBL_NFS_SERVER_CALL_RATE

The number of NFS calls the local machine has processed per second as a NFS server during the interval.

Calls are the system calls used to initiate physical NFS operations. These calls are not always successful due to resource constraints or LAN errors, which means that the call rate could exceed the IO rate. This metric includes both successful and unsuccessful calls.

NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writcache, null and root operations.

GBL_NFS_SERVER_IO

The number of NFS IOs the local machine has completed as an NFS server during the interval. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writcache.

GBL_NFS_SERVER_IO_CUM

The number of NFS IOs the local machine has completed as an NFS server over the cumulative collection time. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writcache.

GBL_NFS_SERVER_IO_PCT

The percentage of NFS IOs the local machine has completed as an NFS server versus total NFS IOs completed during the interval. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

A percentage greater than 50 indicates that this machine is acting more as a server for others. A percentage less than 50 indicates this machine is acting more as a client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writcache.

GBL_NFS_SERVER_IO_PCT_CUM

The percentage of NFS IOs the local machine has completed as an NFS server versus total NFS IOs completed over the cumulative collection time. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

A percentage greater than 50 indicates that this machine is acting more as a server for others. A percentage less than 50 indicates this machine is acting more as a client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_SERVER_IO_RATE

The number of NFS IOs per second the local machine has completed as an NFS server during the interval. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_SERVER_IO_RATE_CUM

The number of NFS IOs per second the local machine has completed as an NFS server over the cumulative collection time. This number represents physical IOs received by the server in contrast to a call which is an attempt to initiate these operations.

Each computer can operate as both a NFS server, and as an NFS client.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS IOs include reads and writes from successful calls to getattr, setattr, lookup, read, readdir, readlink, write, and writocache.

GBL_NFS_SERVER_READ_RATE

The number of NFS "read" operations per second the system processed as an NFS server during the interval.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_SERVER_READ_RATE_CUM

The average number of NFS "read" operations per second the system processed as an NFS server over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS Version 2 read operations consist of getattr, lookup, readlink, readdir, null, root, statfs, and read.

NFS Version 3 read operations consist of getattr, lookup, access, readlink, read, readdir, readdirplus, fsstat, fsinfo, and null.

GBL_NFS_SERVER_WRITE_RATE

The number of NFS "write" operations per second the system processed as an NFS server during the interval.

NFS Version 2 write operations consist of setattr, write, writocache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NFS_SERVER_WRITE_RATE_CUM

The average number of NFS "write" operations per second the system processed as an NFS server over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

NFS Version 2 write operations consist of setattr, write, writocache, create, remove, rename, link, symlink, mkdir, and rmdir.

NFS Version 3 write operations consist of setattr, write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, pathconf, and commit.

GBL_NODENAME

On Unix systems, this is the name of the computer as returned by the command "uname -n" (that is, the string returned from the "hostname" program).

On Windows, this is the name of the computer as returned by GetComputerName.

GBL_NUM_ACTIVE_LS

This indicates the number of LS hosted in a system that are active . If Perf Agent is installed in a guest or in a standalone system this value will be 0.

On Solaris non-global zones, this metric shows value as 0.

GBL_NUM_APP

The number of applications defined in the parm file plus one (for "other").

The application called "other" captures all other processes not defined in the parm file.

You can define up to 999 applications.

GBL_NUM_CPU

The number of physical CPUs on the system. This includes all CPUs, either online or offline. For HP-UX and certain versions of Linux, the sar(1M) command allows you to check the status of the system CPUs. For SUN and DEC, the commands psrinfo(1M) and psradm(1M) allow you to check or change the status of the system CPUs. For AIX, this metric indicates the maximum number of CPUs the system ever had.

On a logical system, this metric indicates the number of virtual CPUs configured. When hardware threads are enabled, this metric indicates the number of logical processors.

On Solaris non-global zones with Uncapped CPUs, this metric shows data from the global zone.

On AIX System WPARs, this metric value is identical to the value on AIX Global Environment.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be "na", some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

GBL_NUM_CPU_CORE

This metric provides the total number of CPU cores on a physical system. On VMs, this metric shows information according to resources available on that VM. On non HP-UX system, this metric is equivalent to active CPU cores. On AIX System WPARs, this metric value is identical to the value on AIX Global Environment. On Windows, this metric will be "na" on Windows Server 2003 Itanium systems.

The Linux kernel currently doesn't provide any metadata information for disabled CPUs. This means that there is no way to find out types, speeds, as well as hardware IDs or any other information that is used to determine the number of cores, the number of threads, the HyperThreading state, etc... If the agent (or Glance) is started while some of the CPUs are disabled, some of these metrics will be "na", some will be based on what is visible at startup time. All information will be updated if/when additional CPUs are enabled and information about them becomes available. The configuration counts will remain at the highest discovered level (i.e. if CPUs are then disabled, the maximum number of CPUs/cores/etc... will remain at the highest observed level). It is recommended that the agent be started with all CPUs enabled.

GBL_NUM_DISK

The number of disks on the system. Only local disk devices are counted in this metric.

On HP-UX, this is a count of the number of disks on the system that have ever had activity over the cumulative collection time.

On Solaris non-global zones, this metric shows value as 0.

On AIX System WPARs, this metric shows value as 0.

GBL_NUM_LS

This indicates the number of LS hosted in a system. If Perf Agent is installed in a guest or in a standalone system this value will be 0.

On Solaris non-global zones, this metric shows value as 0.

GBL_NUM_NETWORK

The number of network interfaces on the system. This includes the loopback interface. On certain platforms, this also include FDDI, Hyperfabric, ATM, Serial Software interfaces such as SLIP or PPP, and Wide Area Network interfaces (WAN) such as ISDN or X.25. The "netstat -i" command also displays the list of network interfaces on the system.

GBL_NUM_SOCKET

The number of physical cpu sockets on the system. On VMs, this metric shows information according to resources available on that VM.

On Windows, this metric will be "na" on Windows Server 2003 Itanium systems.

GBL_NUM_SWAP

The number of configured swap areas.

GBL_NUM_TT

The number of unique Transaction Tracker (TT) transactions that have been registered on this system.

GBL_NUM_USER

The number of users logged in at the time of the interval sample. This is the same as the command "who | wc -l".

For Unix systems, the information for this metric comes from the utmp file which is updated by the login command. For more information, read the man page for utmp. Some applications may create users on the system without using login and updating the utmp file. These users are not reflected in this count.

This metric can be a general indicator of system usage. In a networked environment, however, users may maintain inactive logins on several systems.

On Windows, the information for this metric comes from the Server Sessions counter in the Performance Libraries Server object. It is a count of the number of users using this machine as a file server.

GBL_OSKERNELTYPE

This indicates the word size of the current kernel on the system. Some hardware can load the 64-bit kernel or the 32-bit kernel.

GBL_OSKERNELTYPE_INT

This indicates the word size of the current kernel on the system. Some hardware can load the 64-bit kernel or the 32-bit kernel.

GBL_OSNAME

A string representing the name of the operating system. On Unix systems, this is the same as the output from the "uname -s" command.

GBL_OSRELEASE

The current release of the operating system.

On most Unix systems, this is same as the output from the “uname -r” command.

On AIX, this is the actual patch level of the operating system. This is similar to what is returned by the command “lspp -l bos.rte” as the most recent level of the COMMITTED Base OS Runtime. For example, “5.2.0”.

GBL_OSVERSION

A string representing the version of the operating system. This is the same as the output from the “uname -v” command. This string is limited to 20 characters, and as a result, the complete version name might be truncated.

On Windows, this is a string representing the service pack installed on the operating system.

GBL_PROC_SAMPLE

The number of process data samples that have been averaged into global metrics (such as GBL_ACTIVE_PROC) that are based on process samples.

GBL_RUN_QUEUE

On UNIX systems except Linux, this is the average number of threads waiting in the runqueue over the interval. The average is computed against the number of times the run queue is occupied instead of time. The average is updated by the kernel at a fine grain interval, only when the run queue is occupied. It is not averaged against the interval and can therefore be misleading for long intervals when the run queue is empty most or part of the time. This value matches runq-sz reported by the “sar -q” command. The GBL_LOADAVG* metrics are better indicators of run queue pressure.

On Linux and Windows, this is instantaneous value obtained at the time of logging. On Linux, it shows the number of threads waiting in the runqueue. On Windows, it shows the Processor Queue Length.

On Unix systems, GBL_RUN_QUEUE will typically be a small number. Larger than normal values for this metric indicate CPU contention among threads. This CPU bottleneck is also normally indicated by 100 percent GBL_CPU_TOTAL_UTIL. It may be OK to have GBL_CPU_TOTAL_UTIL be 100 percent if no other threads are waiting for the CPU. However, if GBL_CPU_TOTAL_UTIL is 100 percent and GBL_RUN_QUEUE is greater than the number of processors, it indicates a CPU bottleneck.

On Windows, the Processor Queue reflects a count of process threads which are ready to execute. A thread is ready to execute (in the Ready state) when the only resource it is waiting on is the processor. The Windows operating system itself has many system threads which intermittently use small amounts of processor time. Several low priority threads intermittently wake up and execute for very short intervals. Depending on when the collection process samples this queue, there may be none or several of these low-priority threads trying to execute. Therefore, even on an otherwise quiescent system, the Processor Queue Length can be high. High values for this metric during intervals where the overall CPU utilization (gbl_cpu_total_util) is low do not indicate a performance bottleneck. Relatively high values for this metric during intervals where the overall CPU utilization is near 100% can indicate a CPU performance bottleneck.

HP-UX RUN/PRI/CPU Queue differences for multi-cpu systems:

For example, let's assume we're using a system with eight processors. We start eight CPU intensive threads that consume almost all of the CPU resources. The approximate values shown for the CPU related queue metrics would be:

```
GBL_RUN_QUEUE = 1.0
GBL_PRI_QUEUE = 0.1
GBL_CPU_QUEUE = 1.0
```

Assume we start an additional eight CPU intensive threads. The approximate values now shown are:

```
GBL_RUN_QUEUE = 2.0
GBL_PRI_QUEUE = 8.0
GBL_CPU_QUEUE = 16.0
```

At this point, we have sixteen CPU intensive threads running on the eight processors. Keeping the definitions of the three queue metrics in mind, the run queue is 2 (that is, 16 / 8); the pri queue is 8 (only half of the threads can be active at any given time); and the cpu queue is 16 (half of the threads waiting in the cpu queue that are ready to run, plus one for each active thread).

This illustrates that the run queue is the average of number of threads waiting in the runqueue for all processors; the pri queue is the number of threads that are blocked on "PRI" (priority); and the cpu queue is the number of threads in the cpu queue that are ready to run, including the threads using the CPU.

On Solaris non-global zones, this metric shows data from the global zone.

GBL_RUN_QUEUE_CUM

On UNIX systems except Linux, this is the average number of threads waiting in the runqueue over the cumulative collection time.

On Linux, this is approximately the number of threads waiting in the runqueue over the cumulative collection time.

On Windows, this is approximately the average Processor Queue Length over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

In this case, this metric is a cumulative average of data that was collected as an average. This metric is derived from GBL_RUN_QUEUE.

HP-UX RUN/PRI/CPU Queue differences for multi-cpu systems:

For example, let's assume we're using a system with eight processors. We start eight CPU intensive threads that consume almost all of the CPU resources. The approximate values shown for the CPU related queue metrics would be:

```
GBL_RUN_QUEUE = 1.0
GBL_PRI_QUEUE = 0.1
GBL_CPU_QUEUE = 1.0
```

Assume we start an additional eight CPU intensive threads. The approximate values now shown are:

```
GBL_RUN_QUEUE = 2.0
GBL_PRI_QUEUE = 8.0
GBL_CPU_QUEUE = 16.0
```

At this point, we have sixteen CPU intensive threads running on the eight processors. Keeping the definitions of the three queue metrics in mind, the run queue is 2 (that is, 16 / 8); the pri queue is 8 (only half of the threads can be active at any given time); and the cpu queue is 16 (half of the threads waiting in the cpu queue that are ready to run, plus one for each active thread).

This illustrates that the run queue is the average of number of threads waiting in the runqueue for all processors; the pri queue is the number of threads that are blocked on "PRI" (priority); and the cpu queue is the number of threads in the cpu queue that are ready to run, including the threads using the CPU.

GBL_RUN_QUEUE_HIGH

On UNIX systems except Linux, this is the highest value of average number of threads waiting in the runqueue during any interval over the cumulative collection time.

On Linux, this is the highest value of number of threads waiting in the runqueue during any interval over the cumulative collection time.

GBL_SAMPLE

The number of data samples (intervals) that have occurred over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

GBL_SERIALNO

On HP-UX, this is the ID number of the computer as returned by the command "uname -i". If this value is not available, an empty string is returned.

On SUN, this is the ASCII representation of the hardware-specific serial number. This is printed in hexadecimal as presented by the "hostid" command when possible. If that is not possible, the decimal format is provided instead.

On AIX, this is the machine ID number as returned by the command "uname -m". This number has the form xxyyyymmss. For the RISC System/6000, "xx" position is always 00. The "yyyyyy" positions contain the unique ID number for the central processing unit (cpu). While "mm" represents the model number, and "ss" is the submodel number (always 00).

On Linux, this is the ASCII representation of the hardware-specific serial number, as returned by the command "hostid".

GBL_STARTDATE

The date that the collector started.

GBL_STARTED_PROC

The number of processes that started during the interval.

GBL_STARTED_PROC_RATE

The number of processes that started per second during the interval.

GBL_STARTTIME

The time of day that the collector started.

GBL_STATDATE

The date at the end of the interval, based on local time.

GBL_STATTIME

An ASCII string representing the time at the end of the interval, based on local time.

GBL_SWAP_SPACE_AVAIL

The total amount of potential swap space, in MB.

On HP-UX, this is the sum of the device swap areas enabled by the swapon command, the allocated size of any file system swap areas, and the allocated size of pseudo swap in memory if enabled. Note that this is potential swap space. This is the same as (AVAIL: total) as reported by the "swapinfo -mt" command.

On SUN, this is the total amount of swap space available from the physical backing store devices (disks) plus the amount currently available from main memory. This is the same as (used + available) / 1024, reported by the "swap -s" command.

On Linux, this is same as (Swap: total) as reported by the "free -m" command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_SWAP_SPACE_AVAIL_KB

The total amount of potential swap space, in KB.

On HP-UX, this is the sum of the device swap areas enabled by the swapon command, the allocated size of any file system swap areas, and the allocated size of pseudo swap in memory if enabled. Note that this is potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space.

On HP-UX, this is the same as (AVAIL: total) as reported by the "swapinfo -t" command.

On SUN, this is the total amount of swap space available from the physical backing store devices (disks) plus the amount currently available from main memory. This is the same as (used + available) / 1024, reported by the "swap -s" command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_SWAP_SPACE_DEVICE_AVAIL

The amount of swap space configured on disk devices exclusively as swap space (in MB).

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_DEVICE_UTIL

On HP-UX, this is the percentage of device swap space currently in use of the total swap space available. This does not include file system or remote swap space.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space. Consequently, 100 percent utilization on a single device is not always obtainable. The wasted swap space, and the remainder of allocated SWCHUNKs that have not been used is what is reported in the hold field of the /usr/sbin/swapinfo command.

On HP-UX, when compared to the "swapinfo -mt" command results, this is calculated as:

```
Util = ((USED: dev) sum
        / (AVAIL: total)) * 100
```

On SUN, this is the percentage of total system device swap space currently in use. This metric only gives the percentage of swap space used from the available physical swap device space, and does not include the memory that can be used for swap. (On SunOS 5.X, the virtual swap swapfs can allocate swap space from memory.)

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_USED

The amount of swap space used, in MB.

On HP-UX, "Used" indicates written to disk (or locked in memory), rather than reserved. This is the same as (USED: total - reserve) as reported by the "swapinfo -mt" command.

On SUN, "Used" indicates amount written to disk (or locked in memory), rather than reserved. Swap space is reserved (by decrementing a counter) when virtual memory for a program is created. This is the same as (bytes allocated)/1024, reported by the "swap -s" command.

On Linux, this is same as (Swap: used) as reported by the "free -m" command.

On AIX System WPARs, this metric is NA.

On Solaris non-global zones, this metric is N/A. On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SWAP_SPACE_USED_UTIL

This is the percentage of swap space used.

On HP-UX, "Used %" indicates percentage of swap space written to disk (or locked in memory), rather than reserved. This is the same as percentage of ((USED: total - reserve)/total)*100, as reported by the "swapinfo -mt" command.

On SUN, "Used %" indicates percentage of swap space written to disk (or locked in memory), rather than reserved. Swap space is reserved (by decrementing a counter) when virtual memory for a program is created. This is the same as percentage of ((bytes allocated)/total)*100, reported by the "swap -s" command.

On SUN, global swap space is tracked through the operating system. Device swap space is tracked through the devices. For this reason, the amount of swap space used may differ between the global and by-device metrics. Sometimes pages that are marked to be swapped to disk by the operating system are never swapped. The operating system records this as used swap space, but the devices do not, since no physical IOs occur. (Metrics with the prefix "GBL" are global and metrics with the prefix "BYSWP" are by device.)

On Linux, this is same as percentage of ((Swap: used)/total)*100, as reported by the "free -m" command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

GBL_SWAP_SPACE_UTIL

The percent of available swap space that was being used by running processes in the interval.

On Windows, this is the percentage of virtual memory, which is available to user processes, that is in use at the end of the interval. It is not an average over the entire interval. It reflects the ratio of committed memory to the current commit limit. The limit may be increased by the operating system if the paging file is extended. This is the same as $(\text{Committed Bytes} / \text{Commit Limit}) * 100$ when comparing the results to Performance Monitor.

On HP-UX, swap space must be reserved (but not allocated) before virtual memory can be created. If all of available swap is reserved, then no new processes or virtual memory can be created. Swap space locations are actually assigned (used) when a page is actually written to disk or locked in memory (pseudo swap in memory). This is the same as (PCT USED: total) as reported by the "swapinfo -mt" command.

On Unix systems, this metric is a measure of capacity rather than performance. As this metric nears 100 percent, processes are not able to allocate any more memory and new processes may not be able to run. Very low swap utilization values may indicate that too much area has been allocated to swap, and better use of disk space could be made by reallocating some swap partitions to be user filesystems.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

On AIX System WPARs, this metric is NA.

GBL_SWAP_SPACE_UTIL_CUM

The average percentage of available swap space currently in use (has memory belonging to processes paged or swapped out on it) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space. Consequently, 100 percent utilization on a single device is not always obtainable.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SWAP_SPACE_UTIL_HIGH

The highest average percentage of available swap space currently in use (has memory belonging to processes paged or swapped out on it) in any interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, note that available swap is only potential swap space. Since swap is allocated in fixed (SWCHUNK) sizes, not all of this space may actually be usable. For example, on a 61 MB disk using 2 MB swap size allocations, 1 MB remains unusable and is considered wasted space. Consequently, 100 percent utilization on a single device is not always obtainable.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

GBL_SYSTEM_ID

The network node hostname of the system. This is the same as the output from the "uname -n" command.

On Windows, the name obtained from GetComputerName.

GBL_SYSTEM_TYPE

On Unix systems, this is either the model of the system or the instruction set architecture of the system.

On Windows, this is the processor architecture of the system.

GBL_SYSTEM_UPTIME_HOURS

The time, in hours, since the last system reboot.

GBL_SYSTEM_UPTIME_SECONDS

The time, in seconds, since the last system reboot.

GBL_THRESHOLD_PROCCPU

The process CPU threshold specified in the parm file.

GBL_THRESHOLD_PROCDISK

The process disk threshold specified in the parm file.

GBL_THRESHOLD_PROCIO

The process IO threshold specified in the parm file.

GBL_THRESHOLD_PROCMEM

The process memory threshold specified in the parm file.

GBL_TT_OVERFLOW_COUNT

The number of new transactions that could not be measured because the Measurement Processing Daemon's (midaemon) Measurement Performance Database is full. If this happens, the default Measurement Performance Database size is not large enough to hold all of the registered transactions on this system. This can be remedied by stopping and restarting the midaemon process using the -smdvss option to specify a larger Measurement Performance Database size. The current Measurement Performance Database size can be checked using the midaemon -sizes option.

PROC_APP_ID

The ID number of the application to which the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) belonged during the interval.

Application "other" always has an ID of 1. There can be up to 999 user-defined applications, which are defined in the parm file.

PROC_APP_NAME

The application name of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

Processes (or kernel threads, if HP-UX/Linux Kernel 2.6 and above) are assigned into application groups based upon rules in the parm file. If a process does not fit any rules in this file, it is assigned to the application "other."

The rules include decisions based upon pathname, user ID, priority, and so forth. As these values change during the life of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above), it is re-assigned to another application. This re-evaluation is done every measurement interval.

PROC_CHILD_CPU_SYS_MODE_UTIL

The percentage of system time accumulated by this process's children processes during the interval.

On Unix systems, when a process terminates, its CPU counters (user and system) are accumulated in the parent's "children times" counters. This occurs when the parent waits for (or reaps) the child. See `getrusage(2)`. If the process is an orphan process, its parent becomes the `init(1m)` process, and its CPU times will be accumulated to the `init` process upon termination. The `PROC*_CHILD_*` metrics attempt to report these counters in a meaningful way. If these counters were reported unconditionally as they are incremented, they would be misleading. For example, consider a shell process that forks another process and that process accumulates 100 minutes of CPU time. When that process terminates, the shell would report a huge child time utilization for that interval even though it was generally idle, waiting for that child to terminate. The child process was most likely already reported in previous intervals as it used the CPU time, and therefore it would be confusing to report this time in the parent. If, on the other hand, a process was continuously forking short-lived processes during the interval, it would be useful to report the CPU time used by those children processes. The simple algorithm chosen is to only report children times when their total CPU time is less than the process alive interval, and zero otherwise. It is not fool-proof but it generally yields the right results, i.e., if a process reports high child time utilization for several intervals in a row, it could be a runaway forking process. An example of such a runaway process (or "fork bomb") is:

```
while true ; do ps -ef | grep something done
```

Moderate children times are also a useful way to identify daemons that rely on child processes, or, in the case of the init process it may indicate that many short-lived orphan processes are being created.

Note that this metric is only valid at the process level. It reports CPU time of processes forked and does not report on threads created by processes. The PROC*_CHILD* metrics have no meaning at the thread level, therefore the thread metric of the same name, on systems that report per-thread data, will show "na".

PROC_CHILD_CPU_TOTAL_UTIL

The percentage of system + user time accumulated by this process's children processes during the interval.

On Unix systems, when a process terminates, its CPU counters (user and system) are accumulated in the parent's "children times" counters. This occurs when the parent waits for (or reaps) the child. See `getrusage(2)`. If the process is an orphan process, its parent becomes the `init(1m)` process, and its CPU times will be accumulated to the init process upon termination. The PROC*_CHILD_* metrics attempt to report these counters in a meaningful way. If these counters were reported unconditionally as they are incremented, they would be misleading. For example, consider a shell process that forks another process and that process accumulates 100 minutes of CPU time. When that process terminates, the shell would report a huge child time utilization for that interval even though it was generally idle, waiting for that child to terminate. The child process was most likely already reported in previous intervals as it used the CPU time, and therefore it would be confusing to report this time in the parent. If, on the other hand, a process was continuously forking short-lived processes during the interval, it would be useful to report the CPU time used by those children processes. The simple algorithm chosen is to only report children times when their total CPU time is less than the process alive interval, and zero otherwise. It is not fool-proof but it generally yields the right results, i.e., if a process reports high child time utilization for several intervals in a row, it could be a runaway forking process. An example of such a runaway process (or "fork bomb") is:

```
while true ; do ps -ef | grep something done
```

Moderate children times are also a useful way to identify daemons that rely on child processes, or, in the case of the init process it may indicate that many short-lived orphan processes are being created.

Note that this metric is only valid at the process level. It reports CPU time of processes forked and does not report on threads created by processes. The PROC*_CHILD* metrics have no meaning at the thread level, therefore the thread metric of the same name, on systems that report per-thread data, will show "na".

PROC_CHILD_CPU_USER_MODE_UTIL

The percentage of user time accumulated by this process's children processes during the interval.

On Unix systems, when a process terminates, its CPU counters (user and system) are accumulated in the parent's "children times" counters. This occurs when the parent waits for (or reaps) the child. See `getrusage(2)`. If the process is an orphan process, its parent becomes the `init(1m)` process, and its CPU times will be accumulated to the init process upon termination. The PROC*_CHILD_* metrics attempt to report these counters in a meaningful way. If these counters were reported unconditionally as they are incremented, they would be misleading. For example, consider a shell process that forks another process and that process accumulates 100 minutes of CPU time. When that process terminates, the shell would report a huge child time utilization for that interval even though it was generally idle, waiting for that child to terminate. The child process was most likely already reported in previous intervals as it used the CPU time, and therefore it would be confusing to report this time in the parent. If, on the other hand, a process was continuously forking short-lived processes during the interval, it would be useful to report the CPU time used by those children processes. The simple algorithm chosen is to only report children times when their total CPU time is less than the process alive interval, and zero otherwise. It is not fool-proof but it generally yields the right results, i.e., if a process reports high child time utilization for several intervals in a row, it could be a runaway forking process. An example of such a runaway process (or "fork bomb") is:

```
while true ; do ps -ef | grep something done
```

Moderate children times are also a useful way to identify daemons that rely on child processes, or, in the case of the init process it may indicate that many short-lived orphan processes are being created.

Note that this metric is only valid at the process level. It reports CPU time of processes forked and does not report on threads created by processes. The PROC*_CHILD* metrics have no meaning at the thread level, therefore the thread metric of the same name, on systems that report per-thread data, will show "na".

PROC_CPU_ALIVE_SYS_MODE_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) in system mode as a percentage of the time it is alive during the interval. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in `parm` file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in `parm` file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_ALIVE_TOTAL_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) as a percentage of the time it is alive during the interval. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_ALIVE_USER_MODE_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) in user mode as a percentage of the time it is alive during the interval. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_LAST_USED

The ID number of the processor that last ran the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above). For uni-processor systems, this value is always zero.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_CPU_SYS_MODE_TIME

The CPU time in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_TIME_CUM

The CPU time in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the “-ignore_mt” option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perfd) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with “-ignore_mt” by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_UTIL

The percentage of time that the CPU was in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

High system mode CPU utilizations are normal for IO intensive programs. Abnormally high system CPU utilization can indicate that a hardware problem is causing a high interrupt rate. It can also indicate programs that are not using system calls efficiently.

A classic “hung shell” shows up with very high system mode CPU because it gets stuck in a loop doing terminal reads (a system call) to a device that never responds.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HP-UX, If the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_SYS_MODE_UTIL_CUM

The average percentage of time that the CPU was in system mode in the context of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

A process operates in either system mode (also called kernel mode on Unix or privileged mode on Windows) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to `GlancePlus`, if available for the given platform), whichever occurred last.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_TIME

The total CPU time, in seconds, consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On HP-UX, the total CPU time is the sum of the CPU time components for a process or kernel thread, including system, user, context switch, interrupts processing, realtime, and nice utilization values.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in `parm` file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `“-ignore_mt”` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `“-ignore_mt”` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_TIME_CUM

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time. CPU time is in seconds unless otherwise specified.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to `GlancePlus`, if available for the given platform), whichever occurred last.

This is calculated as

```
PROC_CPU_TOTAL_TIME_CUM =  
    PROC_CPU_SYS_MODE_TIME_CUM +  
    PROC_CPU_USER_MODE_TIME_CUM
```

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HP-UX, If the `ignore_mt` flag is set(true) in `parm` file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in `parm` file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `“-ignore_mt”` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `“-ignore_mt”` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_UTIL

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) as a percentage of the total CPU time available during the interval.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On HP-UX, the total CPU utilization is the sum of the CPU utilization components for a process or kernel thread, including system, user, context switch, interrupts processing, realtime, and nice utilization values.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online.

On platforms other than HP-UX, If the `ignore_mt` flag is set(true) in `parm` file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_TOTAL_UTIL_CUM

The total CPU time consumed by a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) as a percentage of the total CPU time available over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to `GlancePlus`, if available for the given platform), whichever occurred last.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On HP-UX, the total CPU utilization is the sum of the CPU utilization components for a process or kernel thread, including system, user, context switch, interrupts processing, realtime, and nice utilization values.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HP-UX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding `ovpa` startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_TIME

The time, in seconds, the process (or kernel threads, if HP-UX/Linux Kernel 2.6 and above) was using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HP-UX, If the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt`

by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_TIME_CUM

The time, in seconds, the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was using the CPU in user mode over the cumulative collection time. collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_UTIL

The percentage of time the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was using the CPU in user mode during the interval.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HP-UX, if the `ignore_mt` flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the `ignore_mt` flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HP-UX, CPU utilization normalization is controlled by the `-ignore_mt` option of the `midaemon(1m)`. To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (`scopeux`, `glance`, `perfd`) must be shut down and the `midaemon` restarted in the desired mode. To start the `midaemon` with `-ignore_mt` by default, this option should be added in the `/etc/rc.config.d/ovpa` control file. Refer to the documentation regarding ovpa startup. Note that, on HP-UX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_CPU_USER_MODE_UTIL_CUM

The average percentage of time the process (or kernel thread, if HP_UX/Linux Kernel 2.6 and above) was using the CPU in user mode over the cumulative collection time.

User CPU is the time spent in user mode at a normal priority, at real-time priority (on HP-UX, AIX, and Windows systems), and at a nice priority.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

Unlike the global and application CPU metrics, process CPU is not averaged over the number of processors on systems with multiple CPUs. Single-threaded processes can use only one CPU at a time and never exceed 100% CPU utilization.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On multi-processor HP-UX systems, processes which have component kernel threads executing simultaneously on different processors could have resource utilization sums over 100%. The maximum percentage is 100% times the number of CPUs online. On platforms other than HPUX, if the ignore_mt flag is set(true) in parm file, this metric will report values normalized against the number of active cores in the system.

If the ignore_mt flag is not set(false) in parm file, this metric will report values normalized against the number of threads in the system.

This flag will be a no-op if Multithreading is turned off.

On HPUX, CPU utilization normalization is controlled by the "-ignore_mt" option of the midaemon(1m). To change normalization from core-based to logical-cpu-based, or vice-versa, all performance components (scopeux, glance, perf) must be shut down and the midaemon restarted in the desired mode. To start the midaemon with "-ignore_mt" by default, this option should be added in the /etc/rc.config.d/ovpa control file. Refer to the documentation regarding ovpa startup. Note that, on HPUX, unlike other platforms, specifying core-based normalization affects CPU, application, process and thread metrics.

PROC_DISK_PHYS_IO_RATE

The average number of physical disk IOs per second made by the process or kernel thread during the interval.

For processes which run for less than the measurement interval, this metric is normalized over the measurement interval. For example, a process ran for 1 second and did 50 IOs during its life. If the measurement interval is 5 seconds, it is reported as having done 10 IOs per second. If the measurement interval is 60 seconds, it is reported as having done 50/60 or 0.83 IOs per second.

"Disk" in this instance refers to any locally attached physical disk drives (that is, "spindles") that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_IO_RATE_CUM

The number of physical disk IOs per second made by the selected process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

“Disk” in this instance refers to any locally attached physical disk drives (that is, “spindles”) that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_READ

The number of physical reads made by (or for) a process or kernel thread during the last interval.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_READ_CUM

The number of physical reads made by (or for) a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

“Disk” refers to a physical drive (that is, “spindle”), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have “na” values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_READ_RATE

The number of physical reads per second made by (or for) a process or kernel thread during the interval.

"Disk" refers to a physical drive (that is, "spindle"), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_WRITE

The number of physical writes made by (or for) a process or kernel thread during the last interval.

"Disk" in this instance refers to any locally attached physical disk drives (that is, "spindles") that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_WRITE_CUM

The number of physical writes made by (or for) a process or kernel thread over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

"Disk" in this instance refers to any locally attached physical disk drives (that is, "spindles") that may hold file systems and/or swap. NFS mounted disks are not included in this list.

On HP-UX, since this value is reported by the drivers, multiple physical requests that have been collapsed to a single physical operation (due to driver IO merging) are only counted once.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_DISK_PHYS_WRITE_RATE

The number of physical writes per second made by (or for) a process or kernel thread during the interval.

"Disk" refers to a physical drive (that is, "spindle"), not a partition on a drive (unless the partition occupies the entire physical disk). NFS mounted disks are not included in this list.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_EUID

The Effective User ID of a process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_FILE_MODE

A text string summarizing the type of open mode:

```
rd/wr  Opened for input & output
read   Opened for input only
write  Opened for output only
```

PROC_FILE_NAME

The path name or identifying information about the open file descriptor. If the path name string exceeds 40 characters in length, the beginning and the end of the path is shown and the middle of the name is replaced by "...".

An attempt is made to obtain the file path name by either searching the current cylinder group to find directory entries that point to the currently opened inode, or by searching the kernel name cache. Since looking up file path names would require high disk overhead, some names may not be resolved. If the path name can not be resolved, a string is returned indicating the type and inode number of the file.

For the string format including an inode number, you may use the ncheck(1M) program to display the file path name relative to the mount point. Sometimes files may be deleted before they are closed. In these cases, the process file table may still have the inode even though the file is not actually present and as a result, ncheck will fail.

PROC_FILE_NUMBER

The file number of the current open file.

PROC_FILE_OPEN

Number of files the current process has remaining open as of the end of the interval.

PROC_FILE_TYPE

A text string describing the type of the current file. This is one of:

block	Block special device
char	Character device
dir	Directory
fifo	A pipe or named pipe
file	Simple file
link	Symbolic file link
other	An unknown file type

PROC_GROUP_ID

On most systems, this is the real group ID number of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above). On AIX, this is the effective group ID number of the process.

On HP-UX, this is the effective group ID number of the process if not in setgid mode.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_GROUP_NAME

The group name (from /etc/group) of a process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

The group identifier is obtained from searching the /etc/passwd file using the user ID (uid) as a key. Therefore, if more than one account is listed in /etc/passwd with the same user ID (uid) field, the first one is used. If no entry can be found for the user ID in /etc/passwd, the group name is the uid number. If no matching entry in /etc/group can be found, the group ID is returned as the group name.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_INTEREST

A string containing the reason(s) why the process or thread is of interest, based on the thresholds specified in the parm file.

An 'A' indicates that the process or thread exceeds the process CPU threshold, computed using the actual time the process or thread was alive during the interval.

A 'C' indicates that the process or thread exceeds the process CPU threshold, computed using the collection interval. Currently, the same CPU threshold is used for both CPU interest reasons.

A 'D' indicates that the process or thread exceeds the process disk IO threshold.

An 'I' indicates that the process or thread exceeds the IO threshold.

An 'M' indicates that the process exceeds the process memory threshold. This interest reason is only meaningful for processes and therefore not shown for threads.

New processes or threads are identified with an 'N', terminated processes or threads are identified with a 'K'.

Note that the parm file 'nonew', 'nokill' and 'shortlived' settings are logging only options and therefore ignored in Glance components.

PROC_INTERVAL

The amount of time in the interval. This is the same value for all processes (and kernel threads, if HP-UX/Linux Kernel 2.6 and above), regardless of whether they were alive for the entire interval.

Note, calculations such as utilizations or rates are calculated using this standardized process interval (PROC_INTERVAL), rather than the actual alive time during the interval (PROC_INTERVAL_ALIVE). Thus, if a process was only alive for 1 second and used the CPU during its entire life (1 second), but the process sample interval was 5 seconds, it would be reported as using 1/5 or 20% CPU utilization, rather than 100% CPU utilization.

PROC_INTERVAL_ALIVE

The number of seconds that the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was alive during the interval. This may be less than the time of the interval if the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was new or died during the interval.

PROC_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, AIX, and OSF1, this differs from PROC_RUN_TIME in that PROC_RUN_TIME may not include all of the first and last sample interval times and PROC_INTERVAL_CUM does.

PROC_IO_BYTE

On HP-UX, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process or kernel thread, either directly or indirectly, during the interval.

On all other systems, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process during the interval. IOs include disk, terminal, tape and network IO.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On SUN, counts in the MB ranges in general can be attributed to disk accesses and counts in the KB ranges can be attributed to terminal IO. This is useful when looking for processes with heavy disk IO activity. This may vary depending on the sample interval length.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IO_BYTE_CUM

On HP-UX, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process or kernel thread, either directly or indirectly, over the cumulative collection time.

On all other systems, this is the total number of physical IO KBs (unless otherwise specified) that was used by this process over the cumulative collection time. IOs include disk, terminal, tape and network IO.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the

resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IO_BYTE_RATE

On HP-UX, this is the number of physical IO KBs per second that was used by this process or kernel thread, either directly or indirectly, during the interval.

On all other systems, this is the number of physical IO KBs per second that was used by this process during the interval. IOs include disk, terminal, tape and network IO.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On SUN, counts in the MB ranges in general can be attributed to disk accesses and counts in the KB ranges can be attributed to terminal IO. This is useful when looking for processes with heavy disk IO activity. This may vary depending on the sample interval length.

Certain types of disk IOs are not counted by AIX at the process level, so they are excluded from this metric.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_IO_BYTE_RATE_CUM

On HP-UX, this is the average number of physical IO KBs per second that was used by this process or kernel thread, either directly or indirectly, over the cumulative collection time.

On all other systems, this is the average number of physical IO KBs per second that was used by this process over the cumulative collection time. IOs include disk, terminal, tape and network IO.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, indirect IOs include paging and deactivation/reactivation activity done by the kernel on behalf of the process or kernel thread. Direct IOs include disk, terminal, tape, and network IO, but exclude all NFS traffic.

On a threaded operating system, such as HP-UX 11.0 and beyond, process usage of a resource is calculated by summing the usage of that resource by its kernel threads. If this metric is reported for a kernel thread, the value is the resource usage by that single kernel thread. If this metric is reported for a process, the value is the sum of the resource usage by all of its kernel threads. Alive kernel threads and kernel threads that have died during the interval are included in the summation.

On SUN, counts in the MB ranges in general can be attributed to disk accesses and counts in the KB ranges can be attributed to terminal IO. This is useful when looking for processes with heavy disk IO activity. This may vary depending on the sample interval length.

Linux release versions vary with regards to the amount of process-level IO statistics that are available. Some kernels instrument only disk IO, while some provide statistics for all devices together (including tty and other devices with disk IO).

When it is available from your specific release of Linux, the PROC_DISK_PHYS* metrics will report pages of disk IO specifically. The PROC_IO* metrics will report the sum of all types of IO including disk IO, in Kilobytes or KB rates. These metrics will have "na" values on kernels that do not support the instrumentation.

For multi-threaded processes, some Linux kernels only report IO statistics for the main thread. In that case, patches are available that will allow the process instrumentation to report the sum of all thread's IOs, and will also enable per-thread reporting.

PROC_MAJOR_FAULT

Number of major page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

On HP-UX, major page faults and minor page faults are a subset of vfauls (virtual faults). Stack and heap accesses can cause vfauls, but do not result in a disk page having to be loaded into memory.

PROC_MAJOR_FAULT_CUM

Number of major page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, major page faults and minor page faults are a subset of vfauls (virtual faults). Stack and heap accesses can cause vfauls, but do not result in a disk page having to be loaded into memory.

PROC_MEM_DATA_VIRT

On SUN, this is the virtual set size (in KB) of the heap memory for this process. Note that heap can reside partially in BSS and partially in the data segment, so its value will not be the same as PROC_REGION_VIRT of the data segment or PROC_REGION_VIRT_DATA, which is the sum of all data segments for the process.

On the other non HP-UX systems, this is the virtual set size (in KB) of the data segment for this process(or kernel thread, if Linux Kernel 2.6 and above).

A value of "na" is displayed when this information is unobtainable.

On AIX, this is the same as the SIZE value reported by "ps v".

On Linux this value is rounded to PAGESIZE.

PROC_MEM_RES

The size (in KB) of resident memory allocated for the process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, the calculation of this metric differs depending on whether this process has used any CPU time since the midaemon process was started. This metric is less accurate and does not include shared memory regions in its calculation when the process has been idle since the midaemon was started.

On HP-UX, for processes that use CPU time subsequent to midaemon startup, the resident memory is calculated as

```
RSS = sum of private region pages +  
      (sum of shared region pages /  
       number of references)
```

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

This value is only updated when a process uses CPU. Thus, under memory pressure, this value may be higher than the actual amount of resident memory for processes which are idle because their memory pages may no longer be resident or the reference count for shared segments may have changed.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

A value of "na" is displayed when this information is unobtainable. This information may not be obtainable for some system (kernel) processes. It may also not be available for <defunct> processes.

On AIX, this is the same as the RSS value shown by "ps v".

On Windows, this is the number of KBs in the working set of this process. The working set includes the memory pages touched recently by the threads of the process. If free memory in the system is above a threshold, then pages are left in the working set even if they are not in use. When free memory falls below a threshold, pages are trimmed from the working set, but not necessarily paged out to disk from memory. If those pages are subsequently referenced, they will be page faulted back into the working set. Therefore, the working set is a general indicator of the memory resident set size of this process, but it will vary depending on the overall status of memory on the system. Note that the size of the working set is often larger than the amount of pagefile space consumed (PROC_MEM_VIRT).

PROC_MEM_RES_HIGH

The largest value of resident memory (in KB) during its lifetime.

See the description for PROC_MEM_RES for details about how resident memory is determined.

A value of "na" is displayed when this information is unobtainable.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_MEM_SHARED_RES

The size (in KB) of resident memory of shared regions only, such as shared text, shared memory, and shared libraries.

On HP-UX, this value is not affected by the reference count. A value of "na" is displayed when this information is unobtainable.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_MEM_STACK_VIRT

Size (in KB) of the stack for this process(or kernel thread, if Linux Kernel 2.6 and above).

On SUN, the stack is initialized to 8K bytes.

On Linux this value is rounded to PAGESIZE.

PROC_MEM_TEXT_VIRT

Size (in KB) of the private text for this process(or kernel thread, if Linux Kernel 2.6 and above).

On AIX, this is the same as the TSIZ field shown by "ps v".

On Linux this value is rounded to PAGESIZE.

PROC_MEM_VIRT

The size (in KB) of virtual memory allocated for the process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, this consists of the sum of the virtual set size of all private memory regions used by this process, plus this process' share of memory regions which are shared by multiple processes. For processes that use CPU time, the value is divided by the reference count for those regions which are shared.

On HP-UX, this metric is less accurate and does not reflect the reference count for shared regions for processes that were started prior to the mdaemon process and have not used any CPU time since the mdaemon was started.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

On all other Unix systems, this consists of private text, private data, private stack and shared memory. The reference count for shared memory is not taken into account, so the value of this metric represents the total virtual size of all regions regardless of the number of processes sharing access.

Note also that lazy swap algorithms, sparse address space malloc calls, and memory-mapped file access can result in large VSS values. On systems that provide Glance memory regions detail reports, the drilldown detail per memory region is useful to understand the nature of memory allocations for the process.

A value of "na" is displayed when this information is unobtainable. This information may not be obtainable for some system (kernel) processes. It may also not be available for <defunct> processes.

On Windows, this is the number of KBs the process has used in the paging file(s). Paging files are used to store pages of memory used by the process, such as local data, that are not contained in other files. Examples of memory pages which are contained in other files include pages storing a program's .EXE and .DLL files. These would not be kept in pagefile space. Thus, often programs will have a memory working set size (PROC_MEM_RES) larger than the size of its pagefile space.

On Linux this value is rounded to PAGESIZE.

PROC_MINOR_FAULT

Number of minor page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) during the interval.

On HP-UX, major page faults and minor page faults are a subset of vfaulsts (virtual faults). Stack and heap accesses can cause vfaulsts, but do not result in a disk page having to be loaded into memory.

PROC_MINOR_FAULT_CUM

Number of minor page faults for this process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, major page faults and minor page faults are a subset of vfaulsts (virtual faults). Stack and heap accesses can cause vfaulsts, but do not result in a disk page having to be loaded into memory.

PROC_NICE_PRI

The nice priority for the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) when it was last dispatched. The value is a bias used to adjust the priority for the process.

On AIX, the nice user value, makes a process less favored than it otherwise would be, has a range of 0-40 with a default value of 20. The value of PUSER is always added to the value of nice to weight the user process down below the range of priorities expected to be in use by system jobs like the scheduler and special wait queues.

On all other Unix systems, the value ranges from 0 to 39. A higher value causes a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) to be dispatched less.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_PAGEFAULT

The number of page faults that occurred during the interval for the process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

PROC_PAGEFAULT_RATE

The number of page faults per second that occurred during the interval for the process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

PROC_PAGEFAULT_RATE_CUM

The average number of page faults per second that occurred over the cumulative collection time for the process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

PROC_PARENT_PROC_ID

The parent process' PID number.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_PRI

On Unix systems, this is the dispatch priority of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) at the end of the interval. The lower the value, the more likely the process is to be dispatched.

On Windows, this is the current base priority of this process.

On HP-UX, whenever the priority is changed for the selected process or kernel thread, the new value will not be reflected until the process or kernel thread is reactivated if it is currently idle (for example, SLEEPing).

On HP-UX, the lower the value, the more the process or kernel thread is likely to be dispatched. Values between zero and 127 are considered to be “real-time” priorities, which the kernel does not adjust. Values above 127 are normal priorities and are modified by the kernel for load balancing. Some special priorities are used in the HP-UX kernel and subsystems for different activities. These values are described in /usr/include/sys/param.h. Priorities less than PZERO 153 are not signalable.

Note that on HP-UX, many network-related programs such as inetd, biod, and rlogind run at priority 154 which is PPIPE. Just because they run at this priority does not mean they are using pipes. By examining the open files, you can determine if a process or kernel thread is using pipes.

For HP-UX 10.0 and later releases, priorities between -32 and -1 can be seen for processes or kernel threads using the Posix Real-time Schedulers. When specifying a Posix priority, the value entered must be in the range from 0 through 31, which the system then remaps to a negative number in the range of -1 through -32. Refer to the rtsched man pages for more information.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

On AIX, values for priority range from 0 to 127. Processes running at priorities less than PZERO (40) are not signalable.

On Windows, the higher the value the more likely the process or thread is to be dispatched. Values for priority range from 0 to 31. Values of 16 and above are considered to be “realtime” priorities. Threads within a process can raise and lower their own base priorities relative to the process's base priority.

PROC_PROC_ARGV1

The first argument (argv[1]) of the process argument list or the second word of the command line, if present. (For kernel threads, if HP-UX/Linux Kernel 2.6 and above this metric returns the value of the associated process). The HP Performance Agent logs the first 32 characters of this metric.

For releases that support the parm file javaarg flag, this metric may not be the first argument. When javaarg=true, the value of this metric is replaced (for java processes only) by the java class or jar name. This can then be useful to construct parm file java application definitions using the argv1= keyword.

PROC_PROC_CMD

The full command line with which the process was initiated. (For kernel threads, if HP-UX/Linux Kernel 2.6 and above this metric returns the value of the associated process).

On HP-UX, the maximum length returned depends upon the version of the OS, but typically up to 1020 characters are available.

On other Unix systems, the maximum length is 4095 characters.

On Linux, if the command string exceeds 4096 characters, the kernel instrumentation may not report any value.

If the command line contains special characters, such as carriage return and tab, these characters will be converted to , , and so on.

PROC_PROC_ID

The process ID number (or PID) of this process(or associated process for kernel threads, if HP-UX/Linux Kernel 2.6 and above) that is used by the kernel to uniquely identify the process. Process numbers are reused, so they only identify a process for its lifetime.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_PROC_NAME

The process(or kernel thread, if HP-UX/Linux Kernel 2.6 and above) program name. It is limited to 16 characters.

On Unix systems, this is derived from the 1st parameter to the exec(2) system call.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

On Windows, the “System Idle Process” is not reported by Perf Agent since Idle is a process that runs to occupy the processors when they are not executing other threads. Idle has one thread per processor.

PROC_REGION_FILENAME

The file path that corresponds to the front store file of a memory region. For text and data regions, this is the name of the program; for shared libraries it is the library name.

Certain "special" names are displayed if there is no actual "front store" for a memory region. These special names correspond to the region type (for example, <stack>). If the name is "<mmap>", then this is a memory region without "front store," created by the system call mmap(2).

If the file format includes an inode number, use the program ncheck (1M) to display the filename relative to the mount point. Sometimes files may be deleted before they are closed. In these cases, the process file table may still have the inode even though the file is not actually present and as a result, ncheck will fail.

PROC_REGION_PRIVATE_SHARED_FLAG

A text indicator of either private memory (Priv) or shared (Shared) for this memory region. Private memory is only being used by the current process. Shared memory is mapped into the address space of other processes.

PROC_REGION_PROT_FLAG

The protection mode of the process memory segment. It represents Read/Write/eXecute permissions in the same way as ls(1) does for files. This metric is available only for regions that have global protection mode. It is not available ("na") for regions that use per-page protection.

PROC_REGION_TYPE

A text name for the type of this memory region. It can be one of the following:

DATA	Data region
LIBDAT	Shared Library data
LIBTXT	Shared Library text
STACK	Stack region
TEXT	Text (that is, code)

On HP-UX, it can also be one of the following:

GRAPH	Frame buffer lock page
IOMAP	IO region (iomap)
MEMMAP	Memory-mapped file, which includes shared libraries (text and data), or memory created by calls to mmap(2)
NULLDR	Null pointer dereference shared page (see below)
RSESTA	Itanium Registered stack engine region
SIGSTK	Signal stack region
UAREA	User Area region
UNKNWN	Region of unknown type

On HP-UX, a whole page is allocated for NULL pointer dereferencing, which is reported as the NULLDR area. If the program is compiled with the "-z" option (which disallows NULL dereferencing), this area is missing. Shared libraries are accessed as memory mapped files, so that the code will show up as "MEMMAP/Shared" and data will show up as "MEMMAP/Priv".

On SUN, it can also be one of the following:

BSS	Static initialized data
MEMMAP	Memory mapped files
NULLDR	Null pointer dereference

shared page (see below).
SHMEM Shared memory
UNKNWN Region of unknown type

On SUN, programs might have an area for NULL pointer dereferencing, which is reported as the NULLDR area. Special segment types that are supported by the kernel that are used for frame buffer devices or other purposes are typed as UNKNWN. The following kernel processes are examples of this: sched, pageout, and fsflush.

On AIX, as of mid-2010, the OS only provides information for text and data.

PROC_REGION_VIRT

The size (in KBs unless otherwise indicated) of the virtual memory occupied by this memory region.

This value is not affected by the reference count.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_ADDRS

The virtual address of this memory region displayed in hexadecimal showing the space and offset of the region.

On HP-UX, this is a 64-bit (96-bit on a 64-bit OS) hexadecimal value indicating the space and space offset of the region.

PROC_REGION_VIRT_DATA

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by data regions of this process. This value is not affected by the reference count since all data regions are private.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_OTHER

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by regions of this process that are not text, data, stack, or shared memory.

This value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_SHMEM

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by shared memory regions of this process.

Note that this memory is shared by other processes and this figure is reported in their metrics also.

This value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

The number of references is a count of the number of attachments to the memory region. Attachments, for shared regions, may come from several processes sharing the same memory, a single process with multiple attachments, or combinations of these.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_STACK

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by stack regions of this process.

Stack regions are always private and will have a reference count of one.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_REGION_VIRT_TEXT

The size (in KBs unless otherwise indicated) of the total virtual memory occupied by text regions of this process. This value is not affected by the reference count.

This metric is specific to the process as a whole and will not change its value. If this metric is used in a glance adviser script, only pick up one value. Do not sum the values since the same value is shown for all regions.

On AIX, as of mid-2010, the OS only provides information for text and data. Other sizes will always be zero. Note also that the total virtual size may not match the sum of the regions due to inconsistencies in the AIX measurement interfaces.

PROC_RUN_TIME

The elapsed time since a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) started, in seconds.

This metric is less than the interval time if the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) was not alive during the entire first or last interval.

On a threaded operating system such as HP-UX 11.0 and beyond, this metric is available for a process or kernel thread.

PROC_STARTTIME

The creation date and time of the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above).

PROC_STATE

A text string summarizing the current state of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above), either:

```
new      This is the first interval
         the process has been
         displayed.
active   Process is continuing.
died     Process expired during
         the interval.
```

PROC_STATE_FLAG

The Unix STATE flag of the process(or kernel thread, if Linux Kernel 2.6 and above) during the interval.

PROC_STOP_REASON

A text string describing what caused the process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above) to stop executing. For example, if the process is waiting for a CPU while higher priority processes are executing, then its block reason is PRI. A complete list of block reasons follows:

String	Reason for Process Block
died	Process terminated during the interval.
new	Process was created (via the exec() system call) during the interval.
NONE	Process is ready to run. It is not apparent that the process is blocked.
OTHER	Waiting for a reason not decipherable by the measurement software.
PRI	Process is on the run queue.
SLEEP	Waiting for an event to complete.
TRACE	Received a signal to stop because parent is tracing this process.
ZOMB	Process has terminated and the parent is not waiting.

PROC_STOP_REASON_FLAG

A numeric value for the stop reason. This is used by scopeux instead of the ASCII string returned by PROC_STOP_REASON in order to conserve space in the log file.

On a threaded operating system, such as HP-UX 11.0 and beyond, this metric represents a kernel thread characteristic. If this metric is reported for a process, the value for its last executing kernel thread is given. For example, if a process has multiple kernel threads and kernel thread one is the last to execute during the interval, the metric value for kernel thread one is assigned to the process.

PROC_THREAD_COUNT

The total number of kernel threads for the current process.

On Linux systems with Kernel 2.5 and below, every thread has its own process ID so this metric will always be 1.

On Solaris systems, this metric reflects the total number of Light Weight Processes (LWPs) associated with the process.

PROC_THREAD_ID

The thread ID number of this kernel thread, used to uniquely identify it. On Linux systems this metric shall be available from Linux Kernel 2.6 onwards.

PROC_TIME

The time the data for the process (or kernel threads, if HP-UX/Linux Kernel 2.6 and above) was collected, in local time.

PROC_TOP_CPU_INDEX

The index of the process which consumed the most CPU during the interval. From this index, the process PID, process name, and CPU utilization can be obtained. (Even for kernel threads if HP-UX/Linux Kernel 2.6 and above this metric returns the index of the process)

This metric is used by the Performance Tools to index into the Data collection interface's internal table. This is not a metric that will be interesting to Tool users.

PROC_TOP_DISK_INDEX

The index of the process which did the most physical IOs during the last interval.

On HP-UX, note that NFS mounted disks are not considered in this calculation.

With this index, the PID, process name, and IOs per second can be obtained.

This metric is used by the Performance Tools to index into the Data collection interface's internal table. This is not a metric that will be interesting to Tool's users.

PROC_TTY

The controlling terminal for a process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above). This field is blank if there is no controlling terminal. On HP-UX, Linux, and AIX, this is the same as the "TTY" field of the ps command.

On all other Unix systems, the controlling terminal name is found by searching the directories provided in the /etc/ttyrchr file. See man page ttyrchr(4) for details. The matching criteria field ("M", "F" or "I" values) of the ttyrchr file is ignored. If a terminal is not found in one of the ttyrchr file directories, the following directories are searched in the order here: "/dev", "/dev/pts", "/dev/term" and "dev/xt". When a match is found in one of the "/dev" subdirectories, "/dev/" is not displayed as part of the terminal name. If no match is found in the directory searches, the major and minor numbers of the controlling terminal are displayed. In most cases, this value is the same as the "TTY" field of the ps command.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_TTY_DEV

The device number of the controlling terminal for a process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above).

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_UID

The real UID (user ID number) of a process(or kernel threads, if HP-UX/Linux Kernel 2.6 and above). This is the UID returned from the getuid system call.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

PROC_USER_NAME

On Unix systems, this is real user name of a process or the login account (from /etc/passwd) of a process (or kernel thread, if HP-UX/Linux Kernel 2.6 and above). If more than one account is listed in /etc/passwd with the same user ID (uid) field, the first one is used. If an account cannot be found that matches the uid field, then the uid number is returned. This would occur if the account was removed after a process was started.

On Windows, this is the process owner account name, without the domain name this account resides in.

On HP-UX, this metric is specific to a process. If this metric is reported for a kernel thread, the value for its associated process is given.

TBL_BUFFER_HEADER_AVAIL

This is the maximum number of headers pointing to buffers in the file system buffer cache.

On HP-UX, this is the configured number, not the maximum number. This can be set by the "nbuf" kernel configuration parameter. nbuf is used to determine the maximum total number of buffers on the system.

On HP-UX, these are used to manage the buffer cache, which is used for all block IO operations. When nbuf is zero, this value depends on the "bufpages" size of memory (see System Administration Tasks manual). A value of "na" indicates either a dynamic buffer cache configuration, or the nbuf kernel parameter has been left unconfigured and

allowed to “float” with the bufpages parameter. This is not a maximum available value in a fixed buffer cache configuration. Instead, it is the initial configured value. The actual number of used buffer headers can grow beyond this initial value.

On SUN, this value is “nbuf”.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

TBL_BUFFER_HEADER_USED

The number of buffer headers currently in use.

On HP-UX, this dynamic value will rarely change once the system boots. During the system bootup, the kernel allocates a large number of buffer headers and the count is likely to stay at that value after the bootup completes. If the value increases beyond the initial boot value, it will not decrease. Buffer headers are allocated in kernel memory, not user memory, and therefore, will not decrease. This value can exceed the available or configured number of buffer headers in a fixed buffer cache configuration.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_HEADER_USED_HIGH

The largest number of buffer headers used in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_HEADER_UTIL

The percentage of buffer headers currently used.

On HP-UX, a value of “na” indicates either a dynamic buffer cache configuration, or the nbuf kernel parameter has been left unconfigured and allowed to “float” with the bufpages parameter.

On SUN, the buffer cache is a memory pool used by the system to cache inode, indirect block and cylinder group related disk accesses. This is different from the traditional concept of a buffer cache that also holds file system data. On Solaris 5.X, as file data is cached, accesses to it show up as virtual memory IOs. File data caching occurs through memory mapping managed by the virtual memory system, not through the buffer cache. The “nbuf” value is dynamic, but it is very hard to create a situation where the memory cache metrics change, since most systems have more than adequate space for inode, indirect block, and cylinder group data caching. This cache is more heavily utilized on NFS file servers.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_BUFFER_HEADER_UTIL_HIGH

The highest percentage of buffer header used in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On HP-UX, a value of "na" indicates either a dynamic buffer cache configuration, or the nbuf kernel parameter has been left unconfigured and allowed to "float" with the bufpages parameter.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_LOCK_AVAIL

The configured number of file or record locks that can be allocated on the system. Files and/or records are locked by calls to lockf(2). On Linux kernel versions 2.4 and above, available file or record locks is a dynamic value which can grow upto max unsigned long.

TBL_FILE_LOCK_USED

The number of file or record locks currently in use. One file can have multiple locks. Files and/or records are locked by calls to lockf(2).

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

On Solaris non-global zones, this metric is N/A.

TBL_FILE_LOCK_USED_HIGH

The highest number of file locks used by the file system in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_LOCK_UTIL

The percentage of configured file or record locks currently in use. On Linux 2.4 and above kernel versions, this may not give correct picture as file or record locks available may change dynamically and can grow upto max unsigned long.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_LOCK_UTIL_HIGH

The highest percentage of configured file or record locks that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_AVAIL

The number of entries in the file table.

On HP-UX and AIX, this is the configured maximum number of the file table entries used by the kernel to manage open file descriptors.

On HP-UX, this is the sum of the "nfile" and "file_pad" values used in kernel generation.

On SUN, this is the number of entries in the file cache. This is a size. All entries are not always in use. The cache size is dynamic. Entries in this cache are used to manage open file descriptors. They are reused as files are closed and new ones are opened. The size of the cache will go up or down in chunks as more or less space is required in the cache.

On AIX, the file table entries are dynamically allocated by the kernel if there is no entry available. These entries are allocated in chunks.

TBL_FILE_TABLE_USED

The number of entries in the file table currently used by file descriptors.

On SUN, this is the number of file cache entries currently used by file descriptors.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_USED_HIGH

The highest number of entries in the file table that is used by file descriptors in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_UTIL

The percentage of file table entries currently used by file descriptors.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_FILE_TABLE_UTIL_HIGH

The highest percentage of entries in the file table used by file descriptors in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_INODE_CACHE_AVAIL

On HP-UX, this is the configured total number of entries for the incore inode tables on the system. For HP-UX releases prior to 11.2x, this value reflects only the HFS inode table. For subsequent HP-UX releases, this value is the sum of inode tables for both HFS and VxFS file systems (ninode plus vxfs_ninode).

On HP-UX, file system directory activity is done through inodes that are stored on disk. The kernel keeps a memory cache of active and recently accessed inodes to reduce disk IOs. When a file is opened through a pathname, the kernel converts the pathname to an inode number and attempts to obtain the inode information from the cache based on the filesystem type. If the inode entry is not in the cache, the inode is read from disk into the inode cache.

On HP-UX, the number of used entries in the inode caches are usually at or near the capacity. This does not necessarily indicate that the configured sizes are too small because the tables may contain recently used inodes and inodes referenced by entries in the directory name lookup cache. When a new inode cache entry is required and a free entry does not exist, inactive entries referenced by the directory name cache are used. If after freeing inode entries only referenced by the directory name cache does not create enough free space, the message "inode: table is full" message may appear on the console. If this occurs, increase the size of the kernel parameter, ninode. Low directory name cache hit ratios may also indicate an underconfigured inode cache.

On HP-UX, the default formula for the ninode size is:

```
ninode = ((nproc+16+maxusers)+32+
          (2*npty)+(4*num_clients))
```

On all other Unix systems, this is the number of entries in the inode cache. This is a size. All entries are not always in use. The cache size is dynamic.

Entries in this cache are reused as files are closed and new ones are opened. The size of the cache will go up or down in chunks as more or less space is required in the cache.

Inodes are used to store information about files within the file system. Every file has at least two inodes associated with it (one for the directory and one for the file itself). The information stored in an inode includes the owners, timestamps, size, and an array of indices used to translate logical block numbers to physical sector numbers. There is a separate inode maintained for every view of a file, so if two processes have the same file open, they both use the same directory inode, but separate inodes for the file.

TBL_INODE_CACHE_HIGH

On HP-UX and OSF1, this is the highest number of inodes that have been used in any one interval over the cumulative collection time.

On HP-UX, file system directory activity is done through inodes that are stored on disk. The kernel keeps a memory cache of active and recently accessed inodes to reduce disk IOs. When a file is opened through a pathname, the kernel converts the pathname to an inode number and attempts to obtain the inode information from the cache based on the filesystem type. If the inode entry is not in the cache, the inode is read from disk into the inode cache.

On HP-UX, the number of used entries in the inode caches are usually at or near the capacity. This does not necessarily indicate that the configured sizes are too small because the tables may contain recently used inodes and inodes referenced by entries in the directory name lookup cache. When a new inode cache entry is required and a free entry does not exist, inactive entries referenced by the directory name cache are used. If after freeing inode entries only referenced by the directory name cache does not create enough free space, the message "inode: table is full" message may appear on the console. If this occurs, increase the size of the kernel parameter, ninode. Low directory name cache hit ratios may also indicate an underconfigured inode cache.

On HP-UX, the default formula for the ninode size is:

```
ninode = ((nproc+16+maxusers)+32+
          (2*npty)+(4*num_clients))
```

On all other Unix systems, this is the largest size of the inode cache in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_INODE_CACHE_USED

The number of inode cache entries currently in use.

On HP-UX, this is the number of "non-free" inodes currently used. Since the inode table contains recently closed inodes as well as open inodes, the table often appears to be fully utilized. When a new entry is needed, one can usually be found by reusing one of the recently closed inode entries.

On HP-UX, file system directory activity is done through inodes that are stored on disk. The kernel keeps a memory cache of active and recently accessed inodes to reduce disk IOs. When a file is opened through a pathname, the kernel converts the pathname to an inode number and attempts to obtain the inode information from the cache based on the filesystem type. If the inode entry is not in the cache, the inode is read from disk into the inode cache.

On HP-UX, the number of used entries in the inode caches are usually at or near the capacity. This does not necessarily indicate that the configured sizes are too small because the tables may contain recently used inodes and inodes referenced by entries in the directory name lookup cache. When a new inode cache entry is required and a free entry does not exist, inactive entries referenced by the directory name cache are used. If after freeing inode entries only referenced by the directory name cache does not create enough free space, the message "inode: table is full" message may appear on the console. If this occurs, increase the size of the kernel parameter, ninode. Low directory name cache hit ratios may also indicate an underconfigured inode cache.

On HP-UX, the default formula for the ninode size is:

```
ninode = ((nproc+16+maxusers)+32+
          (2*npty)+(4*num_clients))
```

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_BUFFER_ACTIVE

The current active total size (in KBs unless otherwise specified) of all IPC message buffers. These buffers are created by msgsnd(2) calls and released by msgrcv(2) calls. This metric only counts the active message queue buffers, which means that a msgsnd(2) call has been made and the msgrcv(2) has not yet been done on the queue entry or a msgrcv(2) call is waiting on a message queue entry.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_BUFFER_AVAIL

The maximum achievable size (in KBs unless otherwise specified) of the message queue buffer pool on the system. Each message queue can contain many buffers which are created whenever a program issues a `msgsnd(2)` call. Each of these buffers is allocated from this buffer pool.

Refer to the `ipcs(1)` man page for more information.

This value is determined by taking the product of the three kernel configuration variables “`msgseg`”, “`msgssz`” and “`msgmni`”. If the value adds up to a value > 2048GB, “o/p” may be reported on some platforms.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See `ipcs(1)` to report on interprocess communication resources.

TBL_MSG_BUFFER_HIGH

The largest size (in KBs unless otherwise specified) of the message queues in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_BUFFER_USED

The current total size (in KBs unless otherwise specified) of all IPC message buffers. These buffers are created by `msgsnd(2)` calls and released by `msgrcv(2)` calls.

On HP-UX and OSF1, this field corresponds to the CBYTES field of the “`ipcs -qo`” command.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_ACTIVE

The number of message queues currently active. A message queue is allocated by a program using the `msgget(2)` call. This metric returns only the entries in the message queue currently active.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_AVAIL

The configured maximum number of message queues that can be allocated on the system. A message queue is allocated by a program using the `msgget(2)` call.

Refer to the `ipcs(1)` man page for more information.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See `ipcs(1)` to report on interprocess communication resources.

TBL_MSG_TABLE_USED

On HP-UX, this is the number of message queues currently in use.

On all other Unix systems, this is the number of message queues that have been built.

A message queue is allocated by a program using the `msgget(2)` call. See `ipcs(1)` to list the message queues.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_UTIL

The percentage of configured message queues currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_MSG_TABLE_UTIL_HIGH

The highest percentage of configured message queues that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_NUM_NFSDS

The number of NFS servers configured. This is the value "nservers" passed to nfsd (the NFS daemon) upon startup. If no value is specified, the default is one. This value determines the maximum number of concurrent NFS requests that the server can handle. See man page for "nfsd".

TBL_SEM_TABLE_ACTIVE

The number of semaphore identifiers currently active. This means that the semaphores are currently locked by processes. Any new process requesting this semaphore is blocked if IPC_NOWAIT flag is not set.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SEM_TABLE_AVAIL

The configured number of semaphore identifiers (sets) that can be allocated on the system.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_SEM_TABLE_USED

On HP-UX, this is the number of semaphore identifiers currently in use.

On all other Unix systems, this is the number of semaphore identifiers that have been built.

A semaphore identifier is allocated by a program using the semget(2) call. See ipcs(1) to list semaphores.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SEM_TABLE_UTIL

The percentage of configured semaphores identifiers currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SEM_TABLE_UTIL_HIGH

The highest percentage of configured semaphore identifiers that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_ACTIVE

The size (in KBs unless otherwise specified) of the shared memory segments that have running processes attached to them. This may be less than the amount of shared memory used on the system because a shared memory segment may exist and not have any process attached to it.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_AVAIL

The maximum achievable size (in MB unless otherwise specified) of the shared memory pool on the system.

This is a theoretical maximum determined by multiplying the configured maximum number of shared memory entries (shmmni) by the maximum size of each shared memory segment (shmmax). Your system may not have enough virtual memory to actually reach this theoretical limit - one cannot allocate more shared memory than the available reserved space configured for virtual memory.

It should be noted that this value does not include any architectural limitations. (For example, on a 32-bit kernel, there is an addressing limit of 1.75 GB.). If the value adds up to a value > 2048TB, "o/f" may be reported on some platforms.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_SHMEM_HIGH

The highest size (in KBs unless otherwise specified) of shared memory used in any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_ACTIVE

The number of shared memory segments that have running processes attached to them. This may be less than the number of shared memory segments that have been allocated.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_AVAIL

The configured number of shared memory segments that can be allocated on the system.

On SUN, the InterProcess Communication facilities are dynamically loadable. If the amount available is zero, this facility was not loaded when data collection began, and its data is not obtainable. The data collector is unable to determine that a facility has been loaded once data collection has started. If you know a new facility has been loaded, restart the data collection, and the data for that facility will be collected. See ipcs(1) to report on interprocess communication resources.

TBL_SHMEM_TABLE_USED

On HP-UX, this is the number of shared memory segments currently in use.

On all other Unix systems, this is the number of shared memory segments that have been built. This includes shared memory segments with no processes attached to them.

A shared memory segment is allocated by a program using the shmget(2) call. Also refer to ipcs(1).

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_UTIL

The percentage of configured shared memory segments currently in use.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_TABLE_UTIL_HIGH

The highest percentage of configured shared memory segments that have been in use during any one interval over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TBL_SHMEM_USED

The size (in KBs unless otherwise specified) of the shared memory segments.

Additionally, it includes memory segments to which no processes are attached. If a shared memory segment has zero attachments, the space may not always be allocated in memory. See `ipcs(1)` to list shared memory segments.

On Unix systems, this metric is updated every 30 seconds or the sampling interval, whichever is greater.

TTBIN_TRANS_COUNT
TT_CLIENT_BIN_TRANS_COUNT

The number of completed transactions in this range during the last interval.

TTBIN_TRANS_COUNT_CUM
TT_CLIENT_BIN_TRANS_COUNT_CUM

The number of completed transactions in this range over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TTBIN_UPPER_RANGE

The upper range (transaction time) for this TT bin.

There are a maximum of nine user-defined transaction response time bins (`TTBIN_UPPER_RANGE`). The last bin, which is not specified in the transaction configuration file (`ttconf.mwc` on Windows or `tt.conf` on UNIX platforms), is the overflow bin and will always have a value of -2 (overflow). Note that the values specified in the transaction configuration file cannot exceed 2147483.6, which is the number of seconds in 24.85 days. If the user specifies any values greater than 2147483.6, the numbers reported for those bins or Service Level Objectives (SLO) will be -2.

TT_ABORT
TT_CLIENT_ABORT

The number of aborted transactions during the last interval for this transaction.

TT_ABORT_CUM
TT_CLIENT_ABORT_CUM

The number of aborted transactions over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_ABORT_WALL_TIME
TT_CLIENT_ABORT_WALL_TIME

The total time, in seconds, of all aborted transactions during the last interval for this transaction.

TT_ABORT_WALL_TIME_CUM
TT_CLIENT_ABORT_WALL_TIME_CUM

The total time, in seconds, of all aborted transactions over the cumulative collection time for this transaction class.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_APPNO

The registered ARM Application/User ID for this transaction class.

TT_APP_NAME

The registered ARM Application name.

TT_CLIENT_ADDRESS
TT_INSTANCE_CLIENT_ADDRESS

The correlator address. This is the address where the child transaction originated.

TT_CLIENT_ADDRESS_FORMAT
TT_INSTANCE_CLIENT_ADDRESS_FORMAT

The correlator address format. This shows the protocol family for the client network address. Refer to the ARM API Guide for the list and description of supported address formats.

TT_CLIENT_CORRELATOR_COUNT

The number of client or child transaction correlators this transaction has started over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_CLIENT_TRAN_ID
TT_INSTANCE_CLIENT_TRAN_ID

A numerical ID that uniquely identifies the transaction class in this correlator.

TT_COUNT
TT_CLIENT_COUNT

The number of completed transactions during the last interval for this transaction.

TT_COUNT_CUM
TT_CLIENT_COUNT_CUM

The number of completed transactions over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_FAILED
TT_CLIENT_FAILED

The number of Failed transactions during the last interval for this transaction name.

TT_FAILED_CUM
TT_CLIENT_FAILED_CUM

The number of failed transactions over the cumulative collection time for this transaction name.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_FAILED_WALL_TIME
TT_CLIENT_FAILED_WALL_TIME

The total time, in seconds, of all failed transactions during the last interval for this transaction name.

TT_FAILED_WALL_TIME_CUM
TT_CLIENT_FAILED_WALL_TIME_CUM

The total time, in seconds, of all failed transactions over the cumulative collection time for this transaction name.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_INFO

The registered ARM Transaction Information for this transaction.

TT_INPROGRESS_COUNT

The number of transactions in progress (started, but not stopped) at the end of the interval for this transaction class.

TT_INSTANCE_ID

A numerical ID that uniquely identifies this transaction instance at the end of the interval.

TT_INSTANCE_PROC_ID

The ID of the process that started or last updated the transaction instance.

TT_INSTANCE_START_TIME

The time this transaction instance started.

TT_INSTANCE_STOP_TIME

The time this transaction instance stopped. If the transaction instance is currently active, the value returned will be -1. It will be shown as "na" in Glance and GPM to indicate that the transaction instance did not stop during the interval.

TT_INSTANCE_THREAD_ID

The ID of the kernel thread that started or last updated the transaction instance.

TT_INSTANCE_UPDATE_COUNT

The number of times this transaction instance called update since the start of this transaction instance.

TT_INSTANCE_UPDATE_TIME

The time this transaction instance last called update. If the transaction instance is currently active, the value returned will be -1. It will be shown as "na" in Glance and GPM to indicate that a call to update did not occur during the interval.

TT_INSTANCE_WALL_TIME

The elapsed time since this transaction instance was started.

TT_INTERVAL

TT_CLIENT_INTERVAL

The amount of time in the collection interval.

TT_INTERVAL_CUM

TT_CLIENT_INTERVAL_CUM

The amount of time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_MEASUREMENT_COUNT

The number of user defined measurements for this transaction class.

TT_NAME

The registered transaction name for this transaction.

TT_SLO_COUNT

TT_CLIENT_SLO_COUNT

The number of completed transactions that violated the defined Service Level Objective (SLO) by exceeding the SLO threshold time during the interval.

TT_SLO_COUNT_CUM

TT_CLIENT_SLO_COUNT_CUM

The number of completed transactions that violated the defined Service Level Objective by exceeding the SLO threshold time over the cumulative collection time.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_SLO_PERCENT

The percentage of transactions which violate service level objectives.

TT_SLO_THRESHOLD

The upper range (transaction time) of the Service Level Objective (SLO) threshold value. This value is used to count the number of transactions that exceed this user-supplied transaction time value.

TT_TRAN_1_MIN_RATE

For this transaction name, the number of completed transactions calculated to a 1 minute rate. For example, if you completed five of these transactions in a 5 minute window, the rate is one transaction per minute.

TT_TRAN_ID

The registered ARM Transaction ID for this transaction class as returned by `arm_getid()`. A unique transaction id is returned for a unique application id (returned by `arm_init`), tran name, and meta data buffer contents.

TT_UID

The registered ARM Transaction User ID for this transaction name.

TT_UNAME

The registered ARM Transaction User Name for this transaction.

If the `arm_init` function has NULL for the `appl_user_id` field, then the user name is blank. Otherwise, if "*" was specified, then the user name is displayed.

For example, to show the user name for the `armsample1` program, use:

```
appl_id = arm_init("armsample1", "*", 0, 0, 0);
```

To ignore the user name for the `armsample1` program, use:

```
appl_id = arm_init("armsample1", NULL, 0, 0, 0);
```


TT_UPDATE
TT_CLIENT_UPDATE

The number of updates during the last interval for this transaction class. This count includes update calls for completed and in progress transactions.

TT_UPDATE_CUM
TT_CLIENT_UPDATE_CUM

The number of updates over the cumulative collection time for this transaction class. This count includes update calls for completed and in progress transactions.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_USER_MEASUREMENT_AVG
TT_INSTANCE_USER_MEASUREMENT_AVG
TT_CLIENT_USER_MEASUREMENT_AVG

If the measurement type is a numeric or a string, this metric returns "na".

If the measurement type is a counter, this metric returns the average counter differences of the transaction or transaction instance during the last interval. The counter value is the difference observed from a counter between the start and the stop (or last update) of a transaction.

If the measurement type is a gauge, this returns the average of the values passed on any ARM call for the transaction or transaction instance during the last interval.

TT_USER_MEASUREMENT_MAX
TT_INSTANCE_USER_MEASUREMENT_MAX
TT_CLIENT_USER_MEASUREMENT_MAX

If the measurement type is a numeric or a string, this metric returns "na".

If the measurement type is a counter, this metric returns the highest measured counter value over the life of the transaction or transaction instance. The counter value is the difference observed from a counter between the start and the stop (or last update) of a transaction.

If the measurement type is a gauge, this metric returns the highest value passed on any ARM call over the life of the transaction or transaction instance.

TT_USER_MEASUREMENT_MIN
TT_INSTANCE_USER_MEASUREMENT_MIN
TT_CLIENT_USER_MEASUREMENT_MIN

If the measurement type is a numeric or a string, this metric returns "na".

If the measurement type is a counter, this metric returns the lowest measured counter value over the life of the transaction or transaction instance. The counter value is the difference observed from a counter between the start and the stop (or last update) of a transaction.

If the measurement type is a gauge, this metric returns the lowest value passed on any ARM call over the life of the transaction or transaction instance.

TT_USER_MEASUREMENT_NAME
TT_INSTANCE_USER_MEASUREMENT_NAME
TT_CLIENT_USER_MEASUREMENT_NAME

The name of the user defined transactional measurement. The length of the string complies with the ARM 2.0 standard, which is 44 characters long (there are 43 usable characters since this is a NULL terminated character string).

TT_USER_MEASUREMENT_STRING1024_VALUE
TT_INSTANCE_USER_MEASUREMENT_STRING1024_VALUE
TT_CLIENT_USER_MEASUREMENT_STRING1024_VALUE

The last value of the user defined measurement of type string 1024. This type is not implemented and the value is always "na".

TT_USER_MEASUREMENT_STRING32_VALUE
TT_INSTANCE_USER_MEASUREMENT_STRING32_VALUE
TT_CLIENT_USER_MEASUREMENT_STRING32_VALUE

The last value of the user defined measurement of type string 32.

TT_USER_MEASUREMENT_TYPE
TT_INSTANCE_USER_MEASUREMENT_TYPE
TT_CLIENT_USER_MEASUREMENT_TYPE

The type of the user defined transactional measurement.

1 = ARM_COUNTER32
2 = ARM_COUNTER64
3 = ARM_CNTRDIVR32
4 = ARM_GAUGE32
5 = ARM_GAUGE64
6 = ARM_GAUGEDIVR32
7 = ARM_NUMERICID32
8 = ARM_NUMERICID64
9 = ARM_STRING8 (max 8 chars)
10 = ARM_STRING32 (max 32 chars)
11 = ARM_STRING1024 (max 1024 char -- not implemented)

TT_USER_MEASUREMENT_VALUE
TT_INSTANCE_USER_MEASUREMENT_VALUE
TT_CLIENT_USER_MEASUREMENT_VALUE

The last value of the user defined measurement of type counter, gauge, numeric ID, or string 8. Both 32 and 64 bit numeric types are returned as 64 bit values.

TT_WALL_TIME
TT_CLIENT_WALL_TIME

The total time, in seconds, of all transactions completed during the last interval for this transaction.

TT_WALL_TIME_CUM
TT_CLIENT_WALL_TIME_CUM

The total time, in seconds, of all transactions completed over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

TT_WALL_TIME_PER_TRAN
TT_CLIENT_WALL_TIME_PER_TRAN

The average transaction time, in seconds, during the last interval for this transaction.

TT_WALL_TIME_PER_TRAN_CUM
TT_CLIENT_WALL_TIME_PER_TRAN_CUM

The average transaction time, in seconds, over the cumulative collection time for this transaction.

The cumulative collection time is defined from the point in time when either: a) the process (or kernel thread, if HP-UX) was first started, or b) the performance tool was first started, or c) the cumulative counters were reset (relevant only to GlancePlus, if available for the given platform), whichever occurred last.

