
900 Series HP 3000 Computer Systems
**MPE/iX Architected
Interface Facility:
Measurement Interface
Reference Manual**



HP Part No. 36392-90001
Printed in U.S.A. 1992

Fourth Edition
E1092

Notice: This document is licensed for use only by software developers and may not be transferred to end-user customers.

Architected Interfaces Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for direct, indirect, special, incidental, or consequential damages in connection with the furnishing or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information that is protected by copyright. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Copyright © 1992 by Hewlett-Packard Company

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013. Rights for non-DoD U.S. Government Departments and agencies are as set forth in FAR 52.227-19 (c) (1,2).

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304 U.S.A.

Restricted Rights Legend

Printing History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time this document was issued. Many product releases do not require changes to the document. Therefore, do not expect a one-to-one correspondence between product releases and document editions.

Edition	Date	Software Version
First Edition	July 1990	A.00.00
Second Edition	December 1990	A.01.00
Third Edition	June 1992	B.40.00
Fourth Edition	October 1992	C.45.00

Preface

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s not based on PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.

MPE/iX Architected Interface Facility: Measurement Interface Reference Manual (36392-90001) is written for the experienced programmer. It is a reference manual that provides information about architected interfaces (AIFs) that can be used to monitor or diagnose the performance of a 900 Series HP 3000 computer system.

A sophisticated level of operating system knowledge is required to utilize the counters.

This manual is organized into the following chapters and appendixes:

- Chapter 1** **Introduction** contains an introductory overview of architected interfaces in general and measurement interface architected interfaces in particular, as well as installation procedures.
- Chapter 2** **Overview of the Measurement Interface** describes the MPE/iX measurement interface (MI) and the counter organization.
- Chapter 3** **Measurement Interface Architected Interfaces** describes the various types of architected interfaces available.
- Chapter 4** **Architected Interface Use** lists data type naming conventions and the Architected Interface Facility error management strategy.
- Chapter 5** **Architected Interface Descriptions** lists the various architected interfaces and their syntax.
- Appendix A** **AIF Status Messages** contains a list of all the error messages returned by measurement interface architected interfaces.
- Appendix B** **AIF Data Structures** contains a list of all the data structures used in the various architected interfaces and item descriptions.

- Appendix C** **Object Class Information** lists the various object classes for the three mappings and their corresponding index into the array.
- Appendix D** **Semaphore Class Information** lists the various semaphore classes and their corresponding index into the array.
- Appendix E** **Global Counters** describes the counters that measure system-wide events and the corresponding item numbers used for **MIGLOBON** and **MIGLOBGET** calls.
- Appendix F** **Process Counters** describes the counters that measure process-specific events and the corresponding item numbers used for **MIPROCON** and **MIPROCGET** calls.
- Appendix G** **I/O Counters** describes the counters that measure I/O-specific events and the corresponding item numbers used for **MIIOON** and **MIIOGET** calls.
- Appendix H** **Examples** illustrates the use of measurement interface architected interfaces.
- Appendix I** **Processor Counters** describes the counters that measure processor-specific events and the corresponding item numbers used for **MIPRCRGET** calls.

Conventions

UPPERCASE	In a syntax statement, commands and keywords are shown in uppercase characters. The characters must be entered in the order shown; however, you can enter the characters in either uppercase or lowercase. For example: <code>COMMAND</code> can be entered as any of the following: <code>command</code> <code>Command</code> <code>COMMAND</code> It cannot, however, be entered as: <code>comm</code> <code>com_mand</code> <code>comamnd</code>
<i>italics</i>	In a syntax statement or an example, a word in italics represents a parameter or argument that you must replace with the actual value. In the following example, you must replace <i>filename</i> with the name of the file: <code>COMMAND <i>filename</i></code>
<i>bold italics</i>	In a syntax statement, a word in bold italics represents a parameter that you must replace with the actual value. In the following example, you must replace <i>filename</i> with the name of the file: <code>COMMAND(<i>filename</i>)</code>
punctuation	In a syntax statement, punctuation characters (other than brackets, braces, vertical bars, and ellipses) must be entered exactly as shown. In the following example, the parentheses and colon must be entered: <code>(<i>filename</i>):(<i>filename</i>)</code>
<u>underlining</u>	Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, <u>yes</u> is the user's response to the prompt: <code>Do you want to continue? >> <u>yes</u></code>
{ }	In a syntax statement, braces enclose required elements. When several elements are stacked within braces, you must select one. In the following example, you must select either ON or OFF : <code>COMMAND { ON OFF }</code>
[]	In a syntax statement, brackets enclose optional elements. In the following example, OPTION can be omitted: <code>COMMAND <i>filename</i> [OPTION]</code> When several elements are stacked within brackets, you can select one or none of the elements. In the following example, you can select OPTION or <i>parameter</i> or neither. The elements cannot be repeated. <code>COMMAND <i>filename</i> [OPTION <i>parameter</i>]</code>

Conventions (continued)

[...] In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear within the immediately preceding pair of brackets or braces. In the example below, you can select *parameter* zero or more times. Each instance of *parameter* must be preceded by a comma:

[, *parameter*] [...]

In the example below, you only use the comma as a delimiter if *parameter* is repeated; no comma is used before the first occurrence of *parameter*:

[*parameter*] [, ...]





| ... | In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select **A**, **AB**, **BA**, or **B**. The elements cannot be repeated.




$\left\{ \begin{array}{l} \mathbf{A} \\ \mathbf{B} \end{array} \right\} | \dots |$

... In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.

Δ In a syntax statement, the space symbol Δ shows a required blank. In the following example, *parameter* and *parameter* must be separated with a blank:

(*parameter*)Δ(*parameter*)

 The symbol  indicates a key on the keyboard. For example,  represents the carriage return key or  represents the shift key.

 *character*  *character* indicates a control character. For example, Y means that you press the control key and the Y key simultaneously.

Contents

1. Introduction	
Intended Use for Architected Interfaces	1-2
Who Uses Architected Interfaces?	1-2
Installing Measurement Interface Architected Interfaces	1-3
INSTMI	1-3
MIINTR	1-3
MIXL	1-4
Linking MIXL at Link Time	1-4
Linking MIXL at Run Time	1-4
Measurement Interface Documentation Files	1-4
Shipping Products That Use Measurement Interface	
Architected Interfaces	1-5
Using INSTMI	1-5
Using AIFGLOBINSTALL	1-6
2. Overview of the Measurement Interface	
Measurement Interface Counter Organization	2-1
What Is an Object Class?	2-2
What Is a Semaphore Class?	2-3
Mapping Object Class Counters	2-3
Unit of Measurement for Counters	2-4
Detecting and Recovering from Counter Rollover	2-4
Tracking Product Changes Using the Version Number	2-5
3. Measurement Interface Architected Interfaces	
Types of Architected Interfaces	3-1
Access Management Architected Interfaces	3-1
User IDs	3-1
What Is the Purpose of User IDs?	3-2
Information Access Architected Interfaces	3-2
System Configuration Information	3-3
Global Counter Information	3-3
Process Counter Information	3-4
I/O Counter Information	3-4
Processor Counter Information	3-4
Utility Architected Interfaces	3-5

4. Architected Interface Programming Considerations	
Data Type Naming Convention	4-1
Data Type Mappings to Languages	4-2
Error Management	4-3
Overall Status	4-3
Item Status	4-3
Verification Item Status	4-4
5. Architected Interface Descriptions	
AIFACCESSOFF	5-2
AIFACCESSION	5-3
AIFGLOBINSTALL	5-4
AIFSCGET	5-5
AIFSCPUT	5-7
AIFTIME	5-18
MIGLOBGET	5-20
MIGLOBOFF	5-22
MIGLOBON	5-23
MIIOGET	5-25
MIIOOFF	5-28
MIIOON	5-30
MIPRCRGET	5-33
MIPROCGET	5-36
MIPROCOFF	5-40
MIPROCON	5-41
A. AIF Status Messages	
B. AIF Data Structures	
C. Object Class Information	
D. Semaphore Class Information	
E. Global Counters	
F. Process Counters	
G. I/O Counters	
H. Examples	
Example 1 - testmiaif1	H-1
Example 2 - testmiaif2	H-8
I. Processor Counters	
Index	

Figures

2-1. The Path of a Counter Versus Time	2-4
--	-----

Tables

4-1. Types Used and Their Mnemonics	4-1
5-1. System Configuration Item Summary	5-10
5-2. System Configuration Items	5-12
5-3. MIIOGET Item Information	5-27
5-4. MIPCSRGET Item Information	5-35
5-5. MIPROCGET Item Information	5-38
A-1. Errors and Warnings Returned	A-1
C-1. Object Class Information for Standard Map	C-2
C-2. Object Class Information for Small Map	C-3
C-3. Object Class Information for Large Map	C-6
D-1. Semaphore Class Information	D-2
E-1. Global Counter Information	E-2
F-1. Process Counter Information	F-2
G-1. I/O Counter Information	G-2
I-1. Processor Counter Information	I-2

Introduction

The MPE/iX Architected Interface Facility provides reliable, high-performance development tools for 900 Series HP 3000 system management suppliers. The MPE/iX Architected Interface Facility provides specialized procedures, architected interfaces (AIFs), for use by software suppliers and internal and external solutions creators. AIFs provide easy and high-performance access, manipulation, or interception of Hewlett-Packard proprietary operating system and subsystem processes.

AIFs are a software layer between non-operating system software and internals, providing controlled access to MPE/iX internal functionality and data structures. AIFs, executing at user privileged mode (PM), provide a window into MPE/iX internal operations.

AIFs do not supply a direct image of MPE/iX internals, but rather abstracts the operating system structures based upon customer usage. For example, a management utility needs to know everything about a specific session but does not need to know the format of the internal structure's contents. This abstraction gives AIF users independence from MPE/iX details and implementation changes.

AIFs do not provide new operating system functionality, but instead provide supported mechanisms for taking advantage of existing MPE/iX functionality and data structures currently available.

Note

AIFs will change to reflect changes to MPE/iX internals.

It is necessary to have either the HP Pascal/iX or HP C/iX compilers, since the only programming languages supported by AIFs are C and Pascal.

Since AIFs are available only for use with the MPE/iX operating system, a 900 Series HP 3000 computer system is required.

Intended Use for Architected Interfaces

Hewlett-Packard provides two layers of programmatic access into the MPE/iX operating system, allowing software suppliers to select the layer that best meets their needs:

- AIFs provide high performance access.
- System intrinsics provide totally secure access.

In the past, although information has been available through intrinsics, software suppliers have used privileged mode to meet performance needs, risking data integrity and system reliability problems possible with the use of privileged mode.

This concern for performance is addressed in the AIF design, which increases performance while minimizing error checking. AIFs are faster and more functional than intrinsics while providing a higher degree of data integrity and system reliability than privileged mode access.

Caution

Any use of privileged mode should be carefully considered because the normal checks and limitations that apply to standard users are bypassed in privileged mode. A privileged mode program can destroy file integrity and the MPE/iX operating system software. Hewlett-Packard will investigate and attempt to resolve problems resulting from the use of privileged mode code. This service, which is not provided under the standard service contract, is available on a time and materials billing basis. Hewlett-Packard will not support, correct, or attend to any modification of the MPE/iX operating system.

Who Uses Architected Interfaces?

The primary audience of the Architected Interface Facility is third party developers outside of Hewlett-Packard. The secondary audience is Hewlett-Packard internal operating system, subsystem, and application developers.

MPE/iX Architected Interface Facility products are available for purchase by any third party developer. Although other audiences can benefit from using AIFs, they must show a sufficient level of technical sophistication and must be eligible for purchase.

Installing Measurement Interface Architected Interfaces

The Architected Interface Facility: Measurement Interface product includes the following files:

- INSTMI.PUB.SYS
- MIINTR.PUB.SYS
- MIXL.PUB.SYS

INSTMI INSTMI is an installation utility that enables you to execute measurement interface AIF code located on the 900 Series HP 3000 computer system using the user ID that INSTMI has installed. INSTMI must be executed on all systems containing code that calls measurement interface AIFs (for example, your application). INSTMI prompts for a user ID from standard input. This unique user ID is assigned to you by Hewlett-Packard at the time of purchase of the Architected Interface Facility: Measurement Interface product.

Note It is strongly recommended that only the user ID provided by Hewlett-Packard be installed.

INSTMI either initially creates or accesses a file named AIFKUF.PUB.SYS. If INSTMI fails to access the file, it returns the error

COULD NOT ACCESS THE FILE AIFKUF.PUB.SYS.

Note It is important that the user does not have another file of the same name already on the system created for a different purpose than required by Architected Interface Facility products. If a user file named AIFKUF.PUB.SYS already exists, INSTMI attempts to open it instead of creating or opening a valid AIFKUF.PUB.SYS file. In case of installation problems, please contact your Hewlett-Packard support representative.

MIINTR MIINTR is a binary file that contains the intrinsic definitions of all measurement interface AIFs. Use MIINTR in your program to declare measurement interface AIFs.

Following is an example of Pascal code that enables the HP Pascal/iX compiler to locate the measurement interface intrinsic file MIINTR:

```
PROGRAM foo;
PROCEDURE intrinsic_foo; INTRINSIC; { Compiler looks for the procedure }
                                     { intrinsic_foo in the intrinsic   }
                                     { file  SYSINTR.PUB.SYS by default.}

$SYSINTR 'MIINTR.PUB.SYS'$          { Switches the intrinsic file     }
PROCEDURE aif_foo; INTRINSIC;        {Compiler looks in MIINTR.PUB.SYS }
begin
end.
```

Following is an example of C code that enables the HP C/iX compiler to locate the measurement interface intrinsic file MIINTR:

```
#pragma intrinsic_file "mintr.pub.sys"
```

MIXL MIXL.PUB.SYS is an executable library (XL) file containing the executable code for the Architected Interface Facility: Measurement Interface product. It is present on all 900 Series HP 3000 computer systems MPE/iX release 3.0 and later. In order to use measurement interface AIFs, you must link the MIXL library file with your program file using options available either at link time or at run time.

Note MIXL.PUB.SYS must be the last library specified in the executable library list.

Linking MIXL at Link Time

You can link MIXL to your program file at link time using the XL= option of the LINK command. For example:

```
:LINKEDIT  
linkedit>LINK FROM=MYPROG; TO=MYPROG;CAP=PM;XL='XL1, ... XLN,MIXL.PUB.SYS'  
linkedit>EXIT  
:
```

Linking MIXL at Run Time

You can link MIXL to your program file at run time, using the XL= option of the RUN command. For example:

```
:RUN MYPROG;XL='XL1, ... XLN,MIXL.PUB.SYS'
```

Measurement Interface Documentation Files

The Architected Interface Facility: Measurement Interface product also provides a separate magnetic tape (density 1600 bpi) that contains documentation files to ease your AIF development efforts:

- | | |
|--------|--|
| DOBJCL | This file contains the documentation for the object class names and their corresponding index into the array for the various mappings. This documentation is also available in appendix C. |
| DSEMCL | This file contains the documentation for the semaphore class names and their corresponding index into the array. This documentation is also available in appendix D. |
| GLEIN | This file contains the counter names for system-wide events and their corresponding item numbers. More detailed documentation of these counters is available in appendix E. |

PREIN	This file contains the counter names for process-specific events and their corresponding item numbers. More detailed documentation of these counters is available in appendix F.
IOEIN	This file contains the counter names for I/O specific events and their corresponding item number. More detailed documentation of these counters is available in appendix G.

Shipping Products That Use Measurement Interface Architected Interfaces

In order to ship code using measurement interface AIFs to customer sites, you must accomplish one of the two following actions:

- Use the `INSTMI` utility when you install your product on a 900 Series HP 3000 computer system.
- Use the `AIFGLOBINSTALL` AIF in your product to programmatically execute the `INSTMI` utility.

Using `INSTMI`

If you want to install your application using `INSTMI`, you must perform the following steps:

1. You must develop the code according to the guidelines specified above.
2. The file `INSTMI` must be made a part of the final product by shipping it along with your code.
3. The installation procedures for your application must include steps to:
 - a. Restore the file `INSTMI` into the target system's `PUB.SYS`.
 - b. Execute the program `INSTMI` to install your user ID onto the target system.
 - c. Purge the file `INSTMI` to ensure security.

You can accomplish Step 2 by redirecting `STDIN` to mask input, thus avoiding any prompts coming to the screen. Masking the output can be accomplished by redirecting `STDLIST`.

Using AIFGLOBINSTALL

AIFGLOBINSTALL is the programmatic equivalent of executing the INSTMI installation utility. AIFGLOBINSTALL must be executed on all systems containing code that calls measurement interface AIFs (for example, your application). It should be executed once per installation. However, it can be executed each time your application is run without side effects. Your application must execute AIFGLOBINSTALL prior to calling any other measurement interface AIFs.

AIFGLOBINSTALL will fail if not enough disk space is located on LDEV 1. If this occurs, you must create additional free space on LDEV 1 before attempting to re-execute code that contains the call to AIFGLOBINSTALL.

Overview of the Measurement Interface

This chapter provides a brief overview of the MPE/iX measurement interface (MI).

Measurement Interface Counter Organization

Counters are the basic building blocks managed by the MI. Counters have a unit of measure of either numeric count or of time. These counters are updated by the operating system in many ways, including:

- Accumulating the number of times an event happened, for example, the number of times that the CPU entered the busy state.
- Holding the largest value-to-date, for example, the maximum number of processes that were in the ready queue.
- Used as a histogram set, where the first counter measures the number of times the semaphore queue length is between 1 and 3, the second counter measures the number of times the semaphore queue length is between 4 and 6, and the third counter measures the number of times the semaphore queue length is more than 7, and so on.
- Holding the time spent for a particular event, for example, the time spent by the CPU in the busy state.

Each counter is given a unique item number. A counter can hold a single value or an array of values. The counters that contain an array of values can either be an object class counter or a semaphore class counter. As such, all counters can be classified into one of these types:

Simple counter	Contains a single value.
Object class counter	Contains an array of values. Each element of the array corresponds to an object class. An object class refers to a collection of similar objects in the system. These counters measure events that relate to objects, for example, the number of page faults. The number of elements can vary depending on the mapping chosen (explained later in this chapter).

Semaphore class counter Contains an array of values. Each element of the array corresponds to a semaphore class. A semaphore class refers to a collection of semaphores used in the system. These counters measure events that relate to semaphores, for example, the length of the semaphore queue. The various semaphore classes and their corresponding position in the array are described in appendix D.

Each counter measures an event that can be one of the following:

- A system-wide (global) event.
- A process event, specific to a pid.
- An I/O event, specific to an ldev.
- A processor event, specific to a processor.

What Is an Object Class?

By definition, an *Object* is a contiguous unit of virtual space. All virtual space is allocated in terms of objects. An object can be any size, from 1 byte to 4 Gbytes (currently 2 Gbytes, due to compiler limitations with 32-bit unsigned integers). All attributes of the virtual space are associated with the object.

These virtual space objects are used in the operating system for various purposes. For example, there is a stack object associated with every process in the system. Because many of these objects exist in the system, object classes were invented to denote similar objects (for example, all the stack objects in the system belong to the same object class).

All measurements associated with an object are monitored through an object class. For example, the number of page faults is not available for each stack object in the system but is collectively denoted through its object class.

What Is a Semaphore Class?

A semaphore is an implementation of the P and V primitives. Synchronization is accomplished by allowing one process to own the semaphore at a time. Because there are many semaphores used in the operating system for synchronization, the *semaphore class* was invented to denote similar semaphores.

All measurements associated with a semaphore are monitored through its Semaphore class. For example, the queue length buckets reflect the queue length of all the semaphores in a semaphore class.

Mapping Object Class Counters

To conserve space, the object class counters can be obtained in three ways, depending on the granularity of information requested. There are 720 object classes in the system. This requires an allocation of 720x4 bytes for each object class counter. Since not all users need this granularity, a collection of object classes is mapped into one. Thus, mapping provides a means to conserve space when not required. The various mappings available are:

Large map	Gives the maximum granularity. There are 720 classes in this level.
Small map	Consists of 33 classes.
Standard map	Consists of 10 classes.

A particular mapping is selected by the first user to enable the measurement interface. All other subsequent users are forced to use the mapping selected by the first user. The various object classes associated with each map, and their corresponding position in the array, are specified in appendix C.

Note

The mapping refers only to object class counters. It has no relation to semaphore classes.

Unit of Measurement for Counters

The unit of measurement for counters is either:

- **Count.** Count is the number of times the event occurred, or the quantity of an event that happened (for example, the number of bytes).
- **Time.** Time is the length of time an event happened, or the time stamp when an event occurred.

The operating system deals largely with ticks. The number of ticks per millisecond is hardware dependent (typically, above 8000 ticks to a millisecond). Since this granularity is not necessary for performance analysis, the counters with an unit of time are rounded by an MI AIF rounding factor. Thus all timer counters have a unit of *MI_Time*.

In order to convert the counter value retrieved through the `MIxxxGET` calls to ticks, the counter values have to be multiplied by the MI AIF rounding factor. If the counter value is to be converted to milliseconds, the counter value in ticks should be divided by the MI AIF tick factor. These factors are available through a call to `AIFSCGET`.

Detecting and Recovering from Counter Rollover

All counters are stored in integers. Some events are more frequent than others, and when they are monitored for a long duration of time, they may exceed the `maxint` value for an integer, and counter rollover occurs. Under these circumstances, the counter wraps around as shown in the following figure:

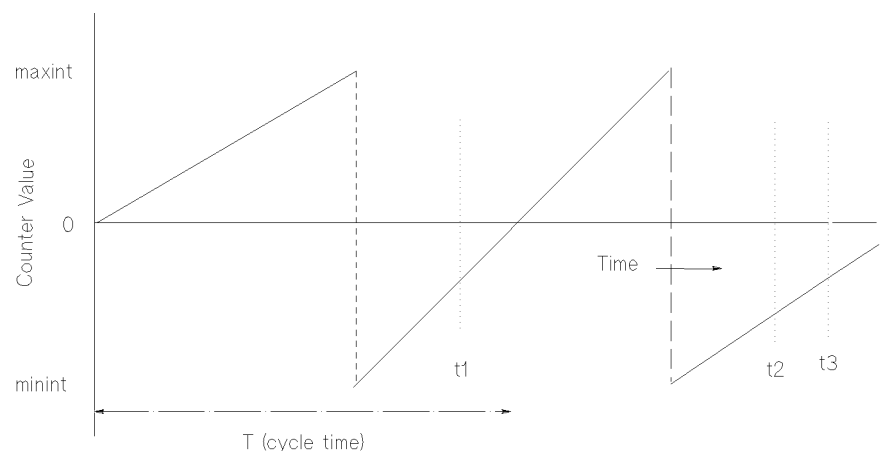


Figure 2-1. The Path of a Counter Versus Time

The figure above has no relation to any existing counter, but is a simple path chosen to illustrate counter rollover. Most counters do not follow a linear path with respect to time. As shown above, the counter falls to `minint` when it reaches `maxint` and then continues to grow until counter rollover is repeated. The cycle time (T) is the time it takes the counter to reach the same value. This cycle time is not fixed, and varies unpredictably over time.

To detect the counter rollover, the samples taken by the tool should be within the cycle time. Since counter rollover is not predictable, the sampling interval can be chosen using trial and error and a reasonable sampling interval should be able to ensure counter rollover detection. Counter rollover has minimal impact upon your development efforts because it affects only a small number of counters (especially timer counters, since they measure time). Under these restrictions, the counter rollover can be detected by checking if the current value of the counter is less than the value obtained from the previous call. Once the counter rollover is detected then it can be recovered as follows:

Let us assume that samples are taken at `t1`, `t2`, and `t3`. Let `x(t)` denote the value of the counter returned at time `t` and let `y(t)` denote the actual value of the counter without counter rollover. Then:

$$y(t1) = x(t1) - \text{minint} + \text{maxint} + 1$$

To calculate the difference between two samples within a cycle use the following formula:

$$y(t2) - y(t1) = x(t2) - \text{minint} + \text{maxint} + 1 - x(t1)$$

Tracking Product Changes Using the Version Number

The Architected Interface Facility: Measurement Interface product version number can be used programmatically to support various versions of the tools. The version number is a character array of eight characters in length, of the format `v.uu.ff` (note that there is a blank space following the `ff`). For example, the initial value of the version number of the first release of the Architected Interface Facility: Measurement Interface product is "A.00.00". The `ff` field is incremented by 1 when enhancements are made to the Architected Interface Facility: Measurement Interface code that does not require any modifications in the code. The `uu` field is incremented when:

- Counters or items have been deleted or added.
- The syntax of the calls have changed.
- The types of certain parameters have changed.

Changes to the uu field may or may not require changes in the tool code. The v field is changed when major changes have been made that might require changes in the tool code.

Measurement Interface Architected Interfaces

This chapter describes the measurement interface architected interfaces (AIFs).

Types of Architected Interfaces

There are two types of measurement interface AIFs:

- Access management AIFs
- Information access AIFs
- Utility AIFs

Access Management Architected Interfaces

Access management AIFs provide a mechanism, the user ID, to validate user access to measurement interface AIFs.

User IDs

Each purchaser of the Architected Interface Facility: Measurement Interface product is assigned a unique user ID. Whenever you call an AIF, you must identify yourself by using your company's user ID. The AIF validates the user ID against a table stored in the measurement interface AIF's internal data area.

Each AIF includes an optional *user_id* parameter. If your program is only going to make a small number of AIF calls, then you'll want to pass the user ID to each AIF as you call it.

However, if your program is going to make a lot of AIF calls, there is a more efficient method to specify your user ID. If your application uses the `AIFACCESSON` AIF to pass your user ID, all subsequent AIF calls made by your application will be assumed to belong to the same user ID. Your user ID is validated once. Use `AIFACCESSOFF` after completing the multiple AIF calls.

Note

You must use the user ID installed through INSTMI in all calls to AIFs described in this manual. If you have purchased another Architected Interface Facility product (for example, operating system AIFs), you still need to call AIFACCESSION with the Architected Interface Facility: Measurement Interface user ID in order to call measurement interface AIFs, even if you have called AIFACCESSION for the other product.

What Is the Purpose of User IDs?

Architected Interface Facility user IDs are used by Hewlett-Packard Response Centers to ensure that AIF-based software products are properly supported. The user IDs are not intended to prevent users who have not purchased an Architected Interface Facility product from calling AIFs; instead, user IDs are intended to guarantee the best possible support.

Because AIFs are trusted procedures, their misuse can cause a number of system problems (including system failures and data corruption). If this should happen, Hewlett-Packard's Response Centers can determine the user IDs associated with any AIF calls that result in errors. In this way, identifying and fixing AIF-related system problems can be accomplished quickly.

Information Access Architected Interfaces

Information access AIFs provide access to performance information while abstracting the structure from the user. There are four types of information accessed by measurement interface AIFs:

- System configuration information
- Global counter information
- Process counter information
- I/O counter information
- Processor counter information

The first information access type listed above, system configuration information, is used to acquire system configuration information required to properly manage and interpret the other three information types (global counter, process counter, and I/O counter information).

Each of the three counter information types (global, process, and I/O) have three AIFs (**MIxxxON**, **MIxxxGET**, and **MIxxxOFF**) that perform the tasks of enabling, acquiring, and disabling counters.

- **MIxxxON** AIFs enable counters that tell the operating system to start measuring the associated events (either global, process, or I/O).
- **MIxxxGET** AIFs return information about counters enabled by the associated **MIxxxON** AIF.
- **MIxxxOFF** AIFs disable the counters enabled by the associated **MIxxxON** AIF, and inform the operating system to stop measuring those events when not used by anyone else.

Note

MIxxxON and **MIxxxGET** AIFs attempt to return as much information as possible each time they are called, returning individual item errors whenever possible. These errors are returned in the *item_status_array* parameter for the items in error, while the rest of the item values are returned normally.

System Configuration Information

System configuration information AIFs are:

- **AIFSCGET**
- **AIFSCPUT**

AIFSCGET and **AIFSCPUT** do not require any keys, because they access system-wide configuration information. **AIFSCGET** provides access to the configuration constants and the dynamically maintained system variables, such as upper limits, but does not provide lists of valid objects on the system. **AIFSCPUT** performs actions similar to the **TUNE** and **ALLOW** commands, as well as some of the startup options.

Some of the configuration constants **AIFSCGET** and **AIFSCPUT** access are the various dispatcher queue priority limits and quanta. The dynamic system information includes the next job/session number to be allocated, the CS average quantum for transactions, and the current **ALLOW** mask.

AIFSCGET is also used to retrieve the roundup factor used for storing timer counters, number of ticks/millisecond, and the Architected Interface Facility: Measurement Interface version ID.

Global Counter Information

The global counter information AIFs are:

- **MIGLOBON**
- **MIGLOBGET**
- **MIGLOBOFF**

These interfaces enable, get, and disable the global counters that measure system-wide events.

Process Counter Information

The process counter information AIFs are:

- MIPROCON
- MIPROCGET
- MIPROCOFF

These interfaces enable, get, and disable process counters that measure specified process events. Process counter events can be monitored for a single process, a collection of processes, or all processes in the system. If all processes are requested at the time of enable, new processes created in the system after enable are also monitored, and information for such processes can be retrieved through MIPROCGET.

When a process dies, counter information for that process is retained until the user calls MIPROCGET for that process, after which no more information is available for that process.

When requesting information for all processes in the system, it is not possible to predetermine the number of processes in the system and, accordingly, to allocate sufficient buffer space. Currently, the measurement interface supports 2700 processes, and the buffer should be able to accommodate information requested for 2700 processes.

Since there is a JSMAIN process for every session and job, and JSMAIN processes are not very active, information on all JSMAIN processes is consolidated into one process. Thus, the counter values for this master JSMAIN process represents the activity of all JSMAIN processes in the system.

I/O Counter Information

The I/O counter information AIFs are:

- MII00N
- MII0GET
- MII0OFF

These interfaces enable, get, and disable device counters that measure specified I/O events. I/O events can be monitored for a single device, a collection of devices, or all devices on the system. Also, you can choose to monitor all devices of the same type (for example, all disk drives or all terminals).

Currently, the maximum logical device number supported is 999. Any other value for ldev is invalid.

Processor Counter Information

The Processor counter information AIF is:

- MIPRCRGET

This interfaces get processor counters that measure specified processor events. Processor events are monitored for all processors when certain global counters are enabled (more specifically when item number 21164 is enabled along with other global counters). Through the MIPRCRGET interface one could request counter values to be returned for a specific processor or for all processors.

Utility Architected Interfaces

AIFTIME converts ticks and microseconds to a more meaningful time, such as date time, clock time, or a string format.

AIFGLOBINSTALL is the programmatic equivalent of executing the INSTMI installation utility.

Architected Interface Programming Considerations

This chapter describes the rules for architected interface (AIF) use, including:

- Data types used to declare AIF parameters.
- Mappings of the data types to programming languages that support AIF use.
- The AIF error management strategy.

Data Type Naming Convention

Below are listed the generic data types (and their mnemonics) that are used to declare the types for the AIF parameters and the values returned.

Table 4-1. Types Used and Their Mnemonics

Mnemonic	Generic Data Type
I32	32-bit signed integer.
U32	32-bit unsigned integer.
B	Boolean.
C	Character.
@32	32-bit address.
@64	64-bit address.
A	Array. Used in combination with other types. For example, CA represents an array of elements, each element containing an ASCII character value; BA represents an array of elements, each element containing a boolean value.
Rec	Record. Refer to appendix B for record structures.

Data Type Mappings to Languages

Most of the information exchange across the AIF interfaces is accomplished through the use of scalar types, which do not require any special treatment. The scalar types include integers, short integers, character arrays, and booleans.

For record types, the documentation provides the Pascal record declaration as well as the packing of the fields as implemented on the HP Pascal/iX compiler. This information should make the call usable from Pascal and C.

For array types, the AIFs allow the user to specify dynamic-length arrays as input. This is done by making the array a part of a simple record declaration. The first field is an integer specifying the number of elements in the array. The second field is the array, with at least as many elements as specified in the first field. Conceptually, it is just an extension of the way strings are implemented on most Pascal compilers today.

Upon return:

- A GET AIF updates the first field to denote the actual number of elements returned.
- A PUT AIF updates the first field to denote the number of elements used.

In addition, if information is truncated in a call to a GET AIF, the AIF returns a warning in the corresponding *item_status_array* element.

Refer to the following manuals for further information on HP Pascal/iX:

HP Pascal Reference Manual (31502-90001)
HP Pascal Programmer's Guide (31502-90002)

Refer to the following manuals for further information on HP C/iX:

HP C/iX Reference Manual (31506-90005)
HP C Programmer's Guide (92434-90002)

Note

If the C programming language is used, all AIF names must be specified in upper case.

Error Management

Error checking has been kept to a minimum for increased performance. Each GET AIF checks to make sure that the specified item exists on the system and that the caller is privileged (HW 2). If this is true, the only checking that is done is to make sure that the addresses that values are being returned into are accessible to the caller. For example, AIFs cannot be used to change the contents of variables in another process's stack.

A PUT AIF also checks to make sure that the specified item exists on the system and that the caller is privileged. Each value that is written to the system is also range-checked whenever possible.

Overall Status

AIFs use the parameter *overall_status* to indicate the status of the call, on the whole.

The data type is `status_type`:

```
status_type = record
    case boolean of
        true : (all : integer);
        false: (info : shortint;
                subsys: shortint);
    end;
```

If an AIF detects no errors, it returns a 32-bit integer with a value of zero. If errors are detected, returns an array of two 16-bit integers. The leftmost 16 bits contain the actual error number, and the rightmost 16 bits contain the subsystem number.

The AIF subsystem number is 516, so AIF errors are reported with a subsystem number of 516. In some cases, the AIFs call another subsystem; if that subsystem detects an error, the called subsystem's number may be returned instead.

The information halfword contains the error message number. Appendix A has a complete list of the AIF error messages. A zero return indicates normal execution. A positive number indicates the index of the last item in the *items_array* that caused an error. There may be more errors, but the user has to walk through the whole *item_status_array* to look for them. A negative number indicates an error condition for the overall call.

Item Status

The parameter called *item_status_array* is an array of `status_type`. (Refer to appendix B.) This array returns status information on each individual item to be transferred. It is returned by the information access procedures and some functionality procedures.

Each element of this array returns a status that follows all the conventions of `status_type`. A status of zero indicates "all OK," a negative number indicates an error, and a positive number indicates warning.

The first few error numbers are the errors encountered in the construction of the item list. They are common to all AIFs. These errors include “Bad item number,” “Bad address of item value,” and “Protection violation.” The other errors are item specific. The statuses referring to items in the GET and PUT arrays return a warning that more elements in the array could have been accommodated.

Note Always initialize the *item_status_array* parameter to zeroes before calling AIFs.

Verification Item Status

There are three optional parameters for a PUT operation:

- *ver_item_nums*
- *ver_items*
- *ver_item_statuses*

These parameters are used to verify that specific conditions are true before the PUT operation takes place. If any of the values in the array passed in the *ver_items* parameter do not match the corresponding item in the *item_array* returned by the previous GET operation, none of the PUT operations take place.

The *ver_item_statuses* parameter is an array of **status_type**. (Refer to appendix B.) This array returns status information on each individual item to be verified. It is returned by the information access PUT AIFs.

Each element of this array returns a status that follows all the conventions of **status_type**. A status of zero indicates “all OK,” a negative number indicates an error and a positive number indicates a warning.

If verification fails, *overall_status* contains an error number -24, indicating that verification failed. It is up to the user to scan the *ver_item_statuses* array to determine which item failed.

Note The *ver_item_statuses* parameter should always be initialized to zeroes before calling AIFs.

Architected Interface Descriptions

This chapter describes measurement interface architected interfaces (AIFs) that can be used for performance monitoring.

AIFACCESSOFF

Deactivates the user ID for the current process.

Syntax

REC	I32
AIFACCESSOFF (<i>overall_status</i> , <i>user_id</i>);	

Parameters	<i>overall_status</i>	record by reference (required)
-------------------	-----------------------	---------------------------------------

Holds the overall status of the call. If an error or warning is returned, the subsys portion of the status record contains a constant for the AIF subsystem. The info portion returns either a negative value, indicating an error; a zero, indicating no error; or a positive value, indicating a warning.

Record type: **status_type** (Refer to appendix B.)

<i>user_id</i>	32-bit signed integer by value (required)
----------------	--

The user ID assigned to a vendor at the time of purchase of an Architected Interface Facility Product. The user ID is used to authorize the current process to call AIFs.

Operation Notes	None.
------------------------	-------

AIFACCESSION

Validates the user ID and authorizes the calling process to call AIFs.

Syntax

```

                                REC      I32
AIFACCESSION ( overall_status, user_id );
```

Parameters*overall_status***record by reference (required)**

Holds the overall status of the call. If an error or warning is returned, the subsys portion of the status contains a constant for the AIF subsystem. The info portion contains either a negative value, indicating an error; a 0, indicating no error; or a positive value, indicating a warning.

Record type: **status_type** (Refer to appendix B.)

*user_id***32-bit signed integer by value (required)**

The user ID assigned to a vendor at the time of purchase of an Architected Interface Facility product. The user ID is used to authorize the current process to call AIFs.

Operation Notes

Each process that makes AIF calls must provide a valid AIF user ID. Once this procedure is called, the process remains authorized until it terminates or until it calls AIFACCESSOFF to deactivate the user ID. Any subsequent AIF calls do require a *user_id* parameter for validation.

If this procedure is not called, the user ID must be passed as a parameter with every AIF call.

AIFGLOBINSTALL

Installs the user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. AIFGLOBINSTALL enables an application to execute measurement interface AIF code located on the target 900 Series HP 3000 computer system.

Syntax

REC	I32
AIFGLOBINSTALL (<i>overall_status</i> , <i>user_id</i>)	

Parameters	<i>overall_status</i>	record by reference (required) Returns the overall status of the call. A zero indicates a successful call. A negative value indicates an error in the overall call. A positive value indicates a warning. Refer to appendix A for meanings of status values. Record type: <code>status_type</code> (Refer to appendix B.)
	<i>user_id</i>	32-bit signed integer by value (required) Passes the user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product.

Operation Notes

AIFGLOBINSTALL is the programmatic equivalent of executing the INSTMI installation utility. AIFGLOBINSTALL (or INSTMI) must be executed on all systems containing code that calls measurement interface AIFs (for example, your application). It should be executed once per installation. However, it can be executed each time your application is run without side effects. Your application must execute AIFGLOBINSTALL prior to calling any other AIFs.

AIFGLOBINSTALL will fail if not enough disk space is located on LDEV 1. If this occurs, you must create additional free space on LDEV 1 before attempting to re-execute code that contains the call to AIFGLOBINSTALL.

AIFSCGET

Allows access to information regarding the current system configuration.

Syntax

```

          REC          I32A          @64A          I32A
AIFSCGET ( overall_status, itemnum_array, item_array, item_status_array,
          I32
          user_id );

```

Parameters*overall_status***record by reference (required)**

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive number indicates an index into *item_status_array*, which indicates the last element signaling an error condition.

Record type: *status_type* (Refer to appendix B.)

*itemnum_array***32-bit signed integer array by reference (required)**

An array of integers where each element is the number of an item to be returned. If there are *n* item numbers being requested, element *n*+1 must be a zero to indicate the end of the element list.

Array type: *itemnum_array_type* (Refer to appendix B.)

*item_array***64-bit address array by reference (required)**

An array of *n* long pointers where each element points to an area of the type appropriate for the corresponding item number in *itemnum_array*. The information to be retrieved is placed in these areas.

Array type: *item_array_type* (Refer to appendix B.)

AIFSCGET

<i>item_status_array</i>	32-bit signed integer array by reference (required) An array of n integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a positive value indicates a warning, and a zero indicates success. Array type: <code>item_status_array_type</code> (Refer to appendix B.)
<i>user_id</i>	32-bit signed integer by value (optional) The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If <i>user_id</i> is passed, it is used to confirm that the caller has the product. If it is not passed, the caller should have previously called AIFACCESSON, or the call fails. Default: 0

Operation Notes

This procedure does not require any specific inputs because the information returned is global to the system.

AIFSCPUT

Allows write access to information about the current system configuration.

Syntax

```

REC          I32A          @64A
AIFSCPUT ( overall_status, itemnum_array, item_array,
          I32A          I32
          item_status_array, user_id,
          I32A          @64A          I32A
          ver_item_nums, ver_items, ver_item_statuses );

```

Parameters	<i>overall_status</i>	<p>record by reference (required)</p> <p>Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates success. This procedure does not require any specific inputs because the information it changes is global to the system.</p> <p>Record type: <code>status_type</code> (Refer to appendix B.)</p>
	<i>itemnum_array</i>	<p>32-bit signed integer array by reference (required)</p> <p>An array of integers where each element is the number of an item to be set. If there are <i>n</i> item numbers being requested, element <i>n</i>+1 must be a zero to indicate the end of the element list.</p> <p>Array type: <code>itemnum_array_type</code> (Refer to appendix B.)</p>
	<i>item_array</i>	<p>64-bit address array by reference (required)</p> <p>An array of <i>n</i> long pointers where each element is a long pointer that points to an area of the type appropriate for the corresponding item number in the <i>itemnum_array</i>. The information to be retrieved is placed in these areas.</p> <p>Array type: <code>item_array_type</code> (Refer to appendix B.)</p>

<i>item_status_array</i>	32-bit signed integer array by reference (required)
	An array of n integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a positive value indicates a warning, and a zero indicates success for that item.
	Array type: <code>item_status_array_type</code> (Refer to appendix B.)
<i>user_id</i>	32-bit signed integer by value (optional)
	The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is passed, it is used to confirm that the caller has the product. If it is not passed, the caller should have previously called AIFACCESSON, or the call will fail.
	Default: 0
<i>ver_item_nums</i>	32-bit signed integer array by reference (optional)
	An array of integers where each element is the number the number of an item to be verified before proceeding with updating. If there are n items being requested, element $n+1$ must be a zero to indicate the end of the element list.
	Array type: <code>itemnum_array_type</code> (Refer to appendix B.)
	Default: nil
<i>ver_items</i>	64-bit address array by reference (optional)
	An array of long pointers where each element points to an area of the type appropriate to the corresponding item number in the <i>ver_item_nums</i> array.
	Array type: <code>item_array_type</code> (Refer to appendix B.)
	Default: nil

ver_item_statuses

**32-bit signed integer array by reference
(optional)**

An array of integers where each element is a returned status corresponding to each *ver_item_nums* array.

Array type: *item_status_array_type* (Refer to appendix B.)

Default: nil

Operation Notes Refer to the following tables for detailed information about system configuration items.

Table 5-1. System Configuration Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
3001	I32	JOB FENCE	Y	Y	0	14	-3005
3002	I32	JOB LIMIT	Y	Y	0	16383	-3006
3003	I32	JOB COUNT	N	Y			
3004	I32	SESSION LIMIT	Y	Y	0	16383	-3007
3005	I32	SESSION COUNT	N	Y			
3006	I32	NEXT JOB #	Y	Y	1	16383	-3008
3007	I32	NEXT SESSION #	Y	Y	1	16383	-3009
3008	I32	JOB SECURITY	Y	Y	0	3	-3010
3009	B	SINGLE USER?	N	Y			
3010	B	OUT OF RES?	N	Y			
3011	B	OUT OF LDEV?	N	Y			
3012	B	LOW ON DISK?	N	Y			
3013	I32	LOGICAL CONSOLE	N	Y			
3014	I32	PHYSIC CONSOLE	N	Y			
3015	BA96	GLOB ALLOW MASK	Y	Y			
3016	BA64	LOGGING MASK	Y	Y			
3017	I32	STREAMS LDEV	N	Y			
3018	I32	SYSTEM OUTFENCE	Y	Y	1	14	-3011
3019	I32	AS QUEUE BASE	N	Y			
3020	I32	AS QUEUE LIMIT	N	Y			
3021	I32	BS QUEUE BASE	N	Y			
3022	I32	BS QUEUE LIMIT	N	Y			
3023	I32	CS QUEUE BASE	Y	Y	127	13567	-3012
3024	I32	CS QUEUE LIMIT	Y	Y	127	13567	-3012
3025	I32	DS QUEUE BASE	Y	Y	127	13567	-3012
3026	I32	DS QUEUE LIMIT	Y	Y	127	13567	-3012
3027	I32	ES QUEUE BASE	Y	Y	127	13567	-3012

Table 5-1. System Configuration Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
3028	I32	ES QUEUE LIMIT	Y	Y	127	13567	-3012
3029	I32	CS QUANTUM MAX	Y	Y			
3030	I32	CS QUANTUM MIN	Y	Y			
3031	I32	DS QUANTUM	Y	Y			
3032	I32	ES QUANTUM	Y	Y			
3033	I32	CS QUANTUM	N	Y			
3034	I32	OPEN FILES/PROC	N	Y			
3035	I32	MAX PROCESSES	N	Y			
3036	I32	MAX JOB/SESSION	N	Y			
3037	CA8	MPE/iX VUF	N	Y			
3038	I32	SERIAL NUMBER	N	Y			
3039	I32	MEMORY SIZE	N	Y			
3040	I32	TOTAL NUM DSTS	N	Y			
3041	I32	AVAIL NUM DSTS	N	Y			
3042	I32	MI ROUND FACTOR	N	Y			
3043	I32	MI TICK/MILLI	N	Y			
3044	CA8	MI VERSION ID	N	Y			
3045	CA8	OS AIF VERSION ID	N	Y			
3046	I32	COLD LOAD ID	N	Y			

Table 5-2. System Configuration Items

Num	Name (Type) and Description
3001	<p>Job Fence (I32) PUT,VERIFY</p> <p>The current jobfence on the system. This should be a value from 0 to 14. If a job's INPRI is higher than the system jobfence, then that job attempts to log on. This can be set using the JOBFENCE command.</p>
3002	<p>Job Limit (I32) PUT,VERIFY</p> <p>The maximum number of jobs that are allowed to be concurrently logged on to the system. This can be set using the LIMIT command and must be a value from 0 to the value configured in SYSGEN.</p>
3003	<p>Job Count (I32) VERIFY</p> <p>The current number of jobs that exist on the system.</p>
3004	<p>Session Limit (I32) PUT,VERIFY</p> <p>The maximum number of sessions that are allowed to be concurrently logged on to the system. This can be set using the LIMIT command and must be a value from 0 to the value configured in SYSGEN.</p>
3005	<p>Session Count (I32) VERIFY</p> <p>The current number of sessions that are logged on the system.</p>
3006	<p>Next Job Number (I32) PUT,VERIFY</p> <p>The number assigned to the next job that is streamed. It should always be a value from 1..16383. Never set this value to a number that could cause two jobs on the machine to have the same job number.</p>
3007	<p>Next Session Number (I32) PUT,VERIFY</p> <p>The number assigned to the next session that successfully logs on. It should always be a value from 1..16383. Never set this value to a number that could cause two sessions on the machine to have the same session number.</p>
3008	<p>Job Security (I32) PUT,VERIFY</p> <p>The current job security with the following values:</p> <ul style="list-style-type: none"> ■ JOB SECURITY HIGH = 0 ■ JOB SECURITY LOW = 3 <p>When job security is high, only the operator logged on to the console can use job control commands. When it is low, users can also issue these commands. It can be set using the JOBSECURITY command.</p>

Table 5-2. System Configuration Items (continued)

Num	Name (Type) and Description																																																																																																												
3009	<p>Single-User Mode (B) VERIFY</p> <p>True when the system is in single-user mode and false when the system is in multiuser mode.</p>																																																																																																												
3010	<p>Out of Resources (B) VERIFY</p> <p>True when a logon has failed because the needed resources were not available. This flag is set back to false when a job or session logs off.</p>																																																																																																												
3011	<p>Out of LDEVs (B) VERIFY</p> <p>True when a job has failed to log on because the specified output device is unavailable. This flag is set to true when that device becomes available.</p>																																																																																																												
3012	<p>Low on Disk Space (B) VERIFY</p> <p>True when there is not enough disk space for a logon to succeed. This flag is set to false when there is enough disk space for a successful logon.</p>																																																																																																												
3013	<p>Logical Console LDEV (I32) VERIFY</p> <p>The current logical console's LDEV number. The logical console LDEV number can be changed using the CONSOLE command.</p>																																																																																																												
3014	<p>Physical Console LDEV (I32) VERIFY</p> <p>The physical console's LDEV number.</p>																																																																																																												
3015	<p>Global ALLOW Mask (BA96) PUT,VERIFY</p> <p>The commands allowed for this system in a packed array of 96 booleans. True indicates an allowed command at these array locations:</p> <table border="0" data-bbox="381 1150 1203 1717"> <tbody> <tr> <td>ABORTIO</td><td>= 1</td> <td>DELETE</td><td>= 19</td> <td>LDISMOUNT</td><td>= 37</td> </tr> <tr> <td>ACCEPT</td><td>= 2</td> <td>DISALLOW</td><td>= 20</td> <td>MRJECONTROL</td><td>= 38</td> </tr> <tr> <td>DOWN</td><td>= 3</td> <td>JOBFENCE</td><td>= 21</td> <td>JOBSECURITY</td><td>= 39</td> </tr> <tr> <td>GIVE</td><td>= 4</td> <td>LIMIT</td><td>= 22</td> <td>DOWNLOAD</td><td>= 40</td> </tr> <tr> <td>HEADOFF</td><td>= 5</td> <td>STOPSPPOOL</td><td>= 23</td> <td>MIOENABLE</td><td>= 41</td> </tr> <tr> <td>HEADON</td><td>= 6</td> <td>SUSPENDSPOOL</td><td>= 24</td> <td>MIODISABLE</td><td>= 42</td> </tr> <tr> <td>REFUSE</td><td>= 7</td> <td>OUTFENCE</td><td>= 25</td> <td>LOG</td><td>= 43</td> </tr> <tr> <td>REPLY</td><td>= 8</td> <td>RECALL</td><td>= 26</td> <td>FOREIGN</td><td>= 44</td> </tr> <tr> <td>STARTSPOOL</td><td>= 9</td> <td>RESUMEJOB</td><td>= 27</td> <td>IMF</td><td>= 45</td> </tr> <tr> <td>TAKE</td><td>= 10</td> <td>RESUMESPOOL</td><td>= 28</td> <td>SHOWCOM</td><td>= 46</td> </tr> <tr> <td>UP</td><td>= 11</td> <td>STREAMS</td><td>= 29</td> <td>OPENQ</td><td>= 47</td> </tr> <tr> <td>MPLINE</td><td>= 12</td> <td>CONSOLE</td><td>= 30</td> <td>SHUTQ</td><td>= 48</td> </tr> <tr> <td>DSCONTROL</td><td>= 13</td> <td>WARN</td><td>= 31</td> <td>DISCSENSING</td><td>= 49</td> </tr> <tr> <td>ABORTJOB</td><td>= 14</td> <td>WELCOME</td><td>= 32</td> <td>VSRESERVESYS</td><td>= 50</td> </tr> <tr> <td>ALLOW</td><td>= 15</td> <td>MON</td><td>= 33</td> <td>VSRELEASESYS</td><td>= 51</td> </tr> <tr> <td>ALTFILE</td><td>= 16</td> <td>MOFF</td><td>= 34</td> <td>VSCLOSE</td><td>= 52</td> </tr> <tr> <td>ALTJOB</td><td>= 17</td> <td>VMOUNT</td><td>= 35</td> <td>VSOPEN</td><td>= 53</td> </tr> <tr> <td>BREAKJOB</td><td>= 18</td> <td>LMOUNT</td><td>= 36</td> <td>unused</td><td>= 54-96</td> </tr> </tbody> </table>	ABORTIO	= 1	DELETE	= 19	LDISMOUNT	= 37	ACCEPT	= 2	DISALLOW	= 20	MRJECONTROL	= 38	DOWN	= 3	JOBFENCE	= 21	JOBSECURITY	= 39	GIVE	= 4	LIMIT	= 22	DOWNLOAD	= 40	HEADOFF	= 5	STOPSPPOOL	= 23	MIOENABLE	= 41	HEADON	= 6	SUSPENDSPOOL	= 24	MIODISABLE	= 42	REFUSE	= 7	OUTFENCE	= 25	LOG	= 43	REPLY	= 8	RECALL	= 26	FOREIGN	= 44	STARTSPOOL	= 9	RESUMEJOB	= 27	IMF	= 45	TAKE	= 10	RESUMESPOOL	= 28	SHOWCOM	= 46	UP	= 11	STREAMS	= 29	OPENQ	= 47	MPLINE	= 12	CONSOLE	= 30	SHUTQ	= 48	DSCONTROL	= 13	WARN	= 31	DISCSENSING	= 49	ABORTJOB	= 14	WELCOME	= 32	VSRESERVESYS	= 50	ALLOW	= 15	MON	= 33	VSRELEASESYS	= 51	ALTFILE	= 16	MOFF	= 34	VSCLOSE	= 52	ALTJOB	= 17	VMOUNT	= 35	VSOPEN	= 53	BREAKJOB	= 18	LMOUNT	= 36	unused	= 54-96
ABORTIO	= 1	DELETE	= 19	LDISMOUNT	= 37																																																																																																								
ACCEPT	= 2	DISALLOW	= 20	MRJECONTROL	= 38																																																																																																								
DOWN	= 3	JOBFENCE	= 21	JOBSECURITY	= 39																																																																																																								
GIVE	= 4	LIMIT	= 22	DOWNLOAD	= 40																																																																																																								
HEADOFF	= 5	STOPSPPOOL	= 23	MIOENABLE	= 41																																																																																																								
HEADON	= 6	SUSPENDSPOOL	= 24	MIODISABLE	= 42																																																																																																								
REFUSE	= 7	OUTFENCE	= 25	LOG	= 43																																																																																																								
REPLY	= 8	RECALL	= 26	FOREIGN	= 44																																																																																																								
STARTSPOOL	= 9	RESUMEJOB	= 27	IMF	= 45																																																																																																								
TAKE	= 10	RESUMESPOOL	= 28	SHOWCOM	= 46																																																																																																								
UP	= 11	STREAMS	= 29	OPENQ	= 47																																																																																																								
MPLINE	= 12	CONSOLE	= 30	SHUTQ	= 48																																																																																																								
DSCONTROL	= 13	WARN	= 31	DISCSENSING	= 49																																																																																																								
ABORTJOB	= 14	WELCOME	= 32	VSRESERVESYS	= 50																																																																																																								
ALLOW	= 15	MON	= 33	VSRELEASESYS	= 51																																																																																																								
ALTFILE	= 16	MOFF	= 34	VSCLOSE	= 52																																																																																																								
ALTJOB	= 17	VMOUNT	= 35	VSOPEN	= 53																																																																																																								
BREAKJOB	= 18	LMOUNT	= 36	unused	= 54-96																																																																																																								

Table 5-2. System Configuration Items (continued)

Num	Name (Type) and Description																																				
3016	<p>System Logging Mask (BA64) PUT,VERIFY</p> <p>The system logging mask in a packed array of 64 booleans. True indicates that the logging type is on. These are the logging types and their array locations:</p> <table border="0" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">LOG FAILURE = 0</td> <td style="width: 50%;">DCE INFORMATION = 18</td> </tr> <tr> <td>SYSTEM UP = 1</td> <td>UNUSED = 19</td> </tr> <tr> <td>JOB INITIATION = 2</td> <td>NCS SPOOLING = 20</td> </tr> <tr> <td>JOB TERMINATION = 3</td> <td>UNUSED = 21-29</td> </tr> <tr> <td>PROCESS TERMINATION = 4</td> <td>AIF = 30</td> </tr> <tr> <td>FILE CLOSE = 5</td> <td>UNUSED = 31-42</td> </tr> <tr> <td>SHUTDOWN = 6</td> <td>CHANGE GROUP = 43</td> </tr> <tr> <td>POWER FAILURE = 7</td> <td>FILE OPEN = 44</td> </tr> <tr> <td>SPOOLING LOG = 8</td> <td>UNUSED = 45</td> </tr> <tr> <td>LINE DISCONNECT = 9</td> <td>MAINT REQ = 46</td> </tr> <tr> <td>LINE CLOSE = 10</td> <td>DCU = 47</td> </tr> <tr> <td>IO ERROR = 11</td> <td>UNUSED = 48-49</td> </tr> <tr> <td>PHYSICAL (DIS)MOUNT = 12</td> <td>DIAGNOSTIC INFORMATION = 50</td> </tr> <tr> <td>LOGICAL (DIS)MOUNT = 13</td> <td>HIGH PRI MACHINE CHECK = 51</td> </tr> <tr> <td>TAPE LABELS = 14</td> <td>LOW PRI MACHINE CHECK = 52</td> </tr> <tr> <td>CONSOLE LOG = 15</td> <td>UNUSED = 53-59</td> </tr> <tr> <td>PROGRAM FILE EVENT = 16</td> <td>CM FCLOSE = 60</td> </tr> <tr> <td>CALL PROGRESS = 17</td> <td>UNUSED = 61-63</td> </tr> </table>	LOG FAILURE = 0	DCE INFORMATION = 18	SYSTEM UP = 1	UNUSED = 19	JOB INITIATION = 2	NCS SPOOLING = 20	JOB TERMINATION = 3	UNUSED = 21-29	PROCESS TERMINATION = 4	AIF = 30	FILE CLOSE = 5	UNUSED = 31-42	SHUTDOWN = 6	CHANGE GROUP = 43	POWER FAILURE = 7	FILE OPEN = 44	SPOOLING LOG = 8	UNUSED = 45	LINE DISCONNECT = 9	MAINT REQ = 46	LINE CLOSE = 10	DCU = 47	IO ERROR = 11	UNUSED = 48-49	PHYSICAL (DIS)MOUNT = 12	DIAGNOSTIC INFORMATION = 50	LOGICAL (DIS)MOUNT = 13	HIGH PRI MACHINE CHECK = 51	TAPE LABELS = 14	LOW PRI MACHINE CHECK = 52	CONSOLE LOG = 15	UNUSED = 53-59	PROGRAM FILE EVENT = 16	CM FCLOSE = 60	CALL PROGRESS = 17	UNUSED = 61-63
LOG FAILURE = 0	DCE INFORMATION = 18																																				
SYSTEM UP = 1	UNUSED = 19																																				
JOB INITIATION = 2	NCS SPOOLING = 20																																				
JOB TERMINATION = 3	UNUSED = 21-29																																				
PROCESS TERMINATION = 4	AIF = 30																																				
FILE CLOSE = 5	UNUSED = 31-42																																				
SHUTDOWN = 6	CHANGE GROUP = 43																																				
POWER FAILURE = 7	FILE OPEN = 44																																				
SPOOLING LOG = 8	UNUSED = 45																																				
LINE DISCONNECT = 9	MAINT REQ = 46																																				
LINE CLOSE = 10	DCU = 47																																				
IO ERROR = 11	UNUSED = 48-49																																				
PHYSICAL (DIS)MOUNT = 12	DIAGNOSTIC INFORMATION = 50																																				
LOGICAL (DIS)MOUNT = 13	HIGH PRI MACHINE CHECK = 51																																				
TAPE LABELS = 14	LOW PRI MACHINE CHECK = 52																																				
CONSOLE LOG = 15	UNUSED = 53-59																																				
PROGRAM FILE EVENT = 16	CM FCLOSE = 60																																				
CALL PROGRESS = 17	UNUSED = 61-63																																				
3017	<p>Streams LDEV (I32) VERIFY</p> <p>The streams LDEV as currently set on the system. The streams LDEV can be set using the STREAMS command.</p>																																				
3018	<p>System Outfence (I32) PUT,VERIFY</p> <p>The system outfence, a value from 1 to 14. A value of 14 prevents all spoolfiles on the system from being printed. This can be set using the OUTFENCE command.</p>																																				
3019	<p>AS Queue Base (I32) VERIFY</p> <p>The base of the AS queue. This value is the highest priority that any process in the AS queue can have. This priority can be mapped to MPE V using this formula:</p> $MPEP_{ri} = (32767 - HPEP_{ri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$																																				
3020	<p>AS Queue Limit (I32) VERIFY</p> <p>The AS queue limit. This value is the lowest priority that any process in the AS queue can have. This priority can be mapped to MPE V using this formula:</p> $MPEP_{ri} = (32767 - HPEP_{ri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$																																				

Table 5-2. System Configuration Items (continued)

Num	Name (Type) and Description
3021	<p>BS Queue Base (I32) VERIFY</p> <p>The base of the BS queue. This value is the highest priority that any process in the BS queue can have. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$
3022	<p>BS Queue Limit (I32) VERIFY</p> <p>The BS queue limit. This value is the lowest priority that any process in the BS queue can have. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$
3023	<p>CS Queue Base (I32) PUT, VERIFY</p> <p>The base of the CS queue. This value is the highest priority that any process in the CS queue can have. It can be set using the TUNE command. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$ <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3024	<p>CS Queue Limit (I32) PUT, VERIFY</p> <p>The CS queue limit. This value is the lowest priority that any process in the CS queue can have. It can be set using the TUNE command. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$ <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3025	<p>DS Queue Base (I32) PUT, VERIFY</p> <p>The base of the DS queue. This value is the highest priority that any process in the DS queue can have. It can be set using the TUNE command. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$ <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3026	<p>DS Queue Limit (I32) PUT, VERIFY</p> <p>The DS queue limit. This value is the lowest priority that any process in the DS queue can have. It can be set using the TUNE command.</p> <p>This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128 \text{ (All formula values are decimal)}$ <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>

Table 5-2. System Configuration Items (continued)

Num	Name (Type) and Description
3027	<p>ES Queue Base (I32) PUT,VERIFY</p> <p>The base of the ES queue. This value is the highest priority that any process in the ES queue can have. It can be set using the TUNE command. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128$ <p>(All formula values are decimal)</p> <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3028	<p>ES Queue Limit (I32) PUT,VERIFY</p> <p>The ES queue limit. This value is the lowest priority that any process in the ES queue can have. It can be set using the TUNE command. This priority can be mapped to MPE V using this formula:</p> $\text{MPEPri} = (32767 - \text{HPEPri}) \text{ DIV } 128$ <p>(All formula values are decimal)</p> <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3029	<p>CS Quantum Maximum (I32) PUT,VERIFY</p> <p>The maximum number of milliseconds that a process in the CS queue may run before its priority starts to decay.</p> <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3030	<p>CS Quantum Minimum (I32) PUT,VERIFY</p> <p>The minimum number of milliseconds that a process in the CS queue must run before its priority may be reduced.</p> <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3031	<p>DS Quantum (I32) PUT,VERIFY</p> <p>The number of milliseconds, which can be set using the TUNE command, that a process in the DS queue may run before its priority starts to decay.</p> <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>
3032	<p>ES Quantum (I32) PUT,VERIFY</p> <p>The number of milliseconds, which can be set using the TUNE command, that a process in the ES queue may run before its priority starts to decay.</p> <p>Any PUT of an item from the range 3023-3032 is not guaranteed to be atomic if the TUNE command gets executed during the PUT.</p>

Table 5-2. System Configuration Items (continued)

Num	Name (Type) and Description
3033	CS Quantum (I32) VERIFY The average number of milliseconds that processes in the CS queue run before they are interrupted. It is used to decide when a process in the CS queue should have its priority decayed. It is maintained dynamically by the system and is very transient in nature.
3034	Maximum Number of Open Files (I32) VERIFY The maximum number of files that a process can have open at the same time.
3035	Maximum Number of Processes (I32) VERIFY The maximum number of processes that can be executing on the machine at the same time.
3036	Maximum number of Jobs/Sessions (I32) VERIFY The maximum Number of jobs and sessions that can be on the machine at the same time.
3037	MPE/iX (CA8) VERIFY The version ID for the MPE/iX operating system on the machine.
3038	Serial Number (I32) VERIFY The machine's serial number. (HPSUSAN)
3039	Memory Size (I32) VERIFY The memory size as the number of 2 K pages.
3040	Total Number of DST entries (I32) VERIFY The total number of DST entries on the machine.
3041	Available Number of DST Entries (I32) VERIFY The total number of DST entries on the machine that are unused.
3042	MI AIF Rounding Factor (I32) VERIFY MI AIF timer counters should be multiplied by this value to convert them to ticks.
3043	MI AIF Tick - Millisecond Conversion Factor (I32) VERIFY The number of ticks in one millisecond for that machine.
3044	MI AIF Version ID (CA8) VERIFY The version of MI AIFs that are on the machine.
3045	OS AIF Version ID (CA8) VERIFY The version of OS AIFs that are on the machine.
3046	Cold Load ID (I32) VERIFY A value that is increased each time the machine is booted.

AIFTIME

This converts ticks or microseconds to meaningful time such as date or clock or in string format.

Syntax

```
REC      REC      REC      REC
AIFTIME ( overall_status, ticks, microsecs, clock,
REC      REC      I32
          date, date_str, user_id );
```

Parameters		
<i>overall_status</i>	record by reference (required)	Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. Record type: <code>status_type</code> (Refer to appendix B.)
<i>ticks</i>	record by reference (optional)	This is the ticks since 1970 which needed to be converted to meaningful time. If this value nor the <code>microsecs</code> is passed then the current time is assumed. Record type: <code>longint_type</code> (Refer to appendix B.)
<i>microsecs</i>	record by reference (optional)	This is the <code>microsecs</code> since 1970 which needed to be converted to meaningful time. If this value nor the <code>microsecs</code> is passed then the current time is assumed. Record type: <code>longint_type</code> (Refer to appendix B.)
<i>clock</i>	record by reference (optional)	This holds the time in hours, minutes, seconds and tens of seconds. Record type: <code>clock_type</code> (Refer to appendix B.)
<i>date</i>	record by reference (optional)	This holds the time in year, month and day of month.

<i>date_str</i>	Record type: <code>date_type</code> (Refer to appendix B.) record by reference (optional) This holds the time in string format for month and day of the week. Record type: <code>date_str_type</code> (Refer to appendix B.)
<i>user_id</i>	32-bit signed integer by value (optional) The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If <i>user_id</i> is passed, it is used to confirm that the caller has the product. If it is not passed, the caller should have previously called AIFACCESSION, or the call fails. Default: 0

MIGLOBGET

Retrieves global counter values enabled by the caller.

Syntax

```
MIGLOBGET ( overall_status, item_nums, buffer_ptr, item_statuses,  
            buffer_size, mapping, user_id );
```

Parameters		
<i>overall_status</i>		record by reference (required) Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than and equal to 20000 is a warning, and a positive value below that indicates an index into the <i>item_status_array</i> , indicating the last element signaling an error condition. Record type: <i>status_type</i> (Refer to appendix B.)
<i>item_nums</i>		32-bit signed integer array by reference (required) An array of integers where each element is the number of an item (counter) to be returned. If there are <i>n</i> item numbers being requested, element <i>n</i> +1 must be zero to indicate the end of the element list.
<i>buffer_ptr</i>		64-bit address by reference (required) This is a long pointer to a buffer which is sufficient to hold all items requested. The size of this buffer should be passed through the <i>buffer_size</i> parameter.
<i>item_statuses</i>		32-bit signed integer array by reference (required) An array of <i>n</i> integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.

<i>buffer_size</i>	32-bit signed integer by value (required) The size of the buffer in bytes.
<i>mapping</i>	32-bit signed integer by value (required) Indicates the mapping for the object class counters requested by the user. The mapping currently in use could be different from the one requested by the user. In such cases, a mapping is done from the one in use to the one requested by the user. Such mapping is possible only from a large set to a smaller set; that is, mapping is possible from a system map of large to a user map of small, and not vice versa. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.
<i>user_id</i>	32-bit signed integer by value (optional) The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSON or the call fails. Default: 0

Operation Notes None.

MIGLOBOFF

Disables the counters previously enabled by the user.

Syntax

REC	I32
MIGLOBOFF (<i>overall_status</i> , <i>user_id</i>);	

Parameters

overall_status

record by reference (required)

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than or equal to 20000 indicates a warning.

Record type: **status_type** (Refer to appendix B.)

user_id

32-bit signed integer by value (optional)

The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSION, or the call fails.

Default: 0

Operation Notes None.

MIGLOBON

Turns on the counters necessary to monitor performance on a system-wide bases.

Syntax

```

REC          I32A          I32A          I32
MIGLOBON ( overall_status, item_nums, item_statuses, mapping,
           I32
           user_id );

```

Parameters*overall_status***record by reference (required)**

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than and equal to 20000 is a warning, and a positive value below that indicates an index into the *item_status_array*, indicating the last element signaling an error condition.

Record type: *status_type* (Refer to appendix B.)

*item_nums***32-bit signed integer array by reference (required)**

An array of integers where each element is the number of an item (counter) to be enabled. If there are *n* item numbers being requested, element *n*+1 must be zero to indicate the end of the element list.

*item_statuses***32-bit signed integer array by reference (required)**

An array of *n* integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.

MIGLOBON

mapping

32-bit signed integer by reference (required)

Indicates the mapping for the object class counters requested by the user. The mapping currently in use is returned which could be the one specified by the user or some other mapping chosen by an earlier MI user. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.

user_id

32-bit signed integer by value (optional)

The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSION, or the call fails.

Default: 0

Operation Notes

In order to monitor the performance on a system-wide basis, the necessary counters should be turned on. This procedure communicates this information to the system, which starts collecting data for the requested counters. In order to select a different set of counters, the current set of counters should be disabled through the call MIGLOBOFF before calling MIGLOBON. This procedure can be called only once without calling MIGLOBOFF, that is, there should be a matching MIGLOBON/MIGLOBOFF before calling MIGLOBON again.

MIIOGET

Retrieves I/O counter values enabled by the caller for the desired logical device(s).

Syntax

```

          REC          I32A          @64          I32A
MIIOGET (overall_status, item_nums, buffer_ptr, item_statuses,
          I32          I32          I32          I32          I32
          buffer_size, mapping, ldev, num_of_ldevs, user_id);
    
```

Parameters

overall_status

record by reference (required)

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than or equal to 20000 is a warning, and a positive value below that indicates an index into the *item_status_array*, indicating the last element signaling an error condition.

Record type: `status_type` (Refer to appendix B.)

item_nums

32-bit signed integer array by reference (required)

An array of integers where each element is the number of an item (counter) to be returned. If there are *n* item numbers being requested, element *n*+1 must be zero to indicate the end of the element list.

buffer_ptr

64-bit address by reference (required)

A long pointer to a buffer, which is sufficient to hold all items requested. The size of this buffer should be passed through the *buffer_size* parameter.

item_statuses

32-bit signed integer array by reference (required)

An array of *n* integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.

<i>buffer_size</i>	32-bit signed by value (required) The size of the buffer in bytes.
<i>mapping</i>	32-bit signed integer by value (required) Indicates the mapping for the object class counters requested by the user. The mapping currently in use may be different from the one requested by the user. In such cases, a mapping is done from the one in use to the one requested by the user. Such mapping is possible only from a large set to a smaller set; that is, mapping is possible from a system map of large to a user map of small, and not vice versa. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.
<i>ldev</i>	32-bit signed integer by reference (optional) The <i>logical device (LDEV)</i> number for which counter values are requested. Specifying a value between -1 and -11 indicates the following: <ul style="list-style-type: none"> -1 For all logical devices enabled by the user. -2 For all disks enabled by the user. -3 For all tapes enabled by the user. -4 For all terminals enabled by the user. -5 For all printers enabled by the user. -6 For all serial printers enabled by the user. -7 For all spooler devices enabled by the user. -8 For all data communication devices enabled by the user. -9 For all DS terminals enabled by the user. -10 For all DS printers enabled by the user. -11 For all user devices enabled by the user. Default: -1
<i>num_of_ldevs</i>	32-bit signed integer by reference (optional) Indicates the number of LDEVs for which information is returned in the buffer.

user_id

32-bit signed integer by value (optional)

The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSON, or the call fails.

Default: 0

Operation Notes

Table 5-3. MIIOGET Item Information

Num	Name (Type) and Description
20020	<p>LDEV (I32)</p> <p>This returns the LDEV number for which the counter values are requested.</p>

MIIOFF

Disables I/O counters enabled by the user.

Syntax

```
REC      I32      I32
MIIOFF ( overall_status, ldev, user_id );
```

Parameters

overall_status

record by reference (required)

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than and equal to 20000 indicates a warning.

Record type: `status_type` (Refer to appendix B.)

ldev

32-bit signed integer by reference (optional)

The logical device (LDEV) number for which the counters are to be disabled. Specifying a value between -1 and -11 indicates the following:

- 1 Disable all logical devices in the system.
- 2 Disable all disks in the system.
- 3 Disable all tapes in the system.
- 4 Disable all terminals in the system.
- 5 Disable all printers in the system.
- 6 Disable all serial printers in the system.
- 7 Disable all spooler devices in the system.
- 8 Disable all data communication devices in the system.
- 9 Disable all DS terminals in the system.
- 10 Disable all DS printers in the system.
- 11 Disable all user devices in the system.

Default: -1

user_id

32-bit signed integer by value (optional)

The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using `AIFACCESSION`, or the call fails.

Default: 0

Operation Notes

When a specific LDEV is disabled, all LDEVs in that device type enabled by the user are disabled. Thus the user cannot disable a certain LDEV and continue to monitor other LDEVs belonging to the same device type.

MIIOON

Turns on I/O related counters to monitor performance on specified logical device(s).

Syntax

```
MIIOON ( REC overall_status, I32A item_nums, I32A item_statuses, I32 mapping,  
I32 ldev, I32 user_id);
```

Parameters

overall_status

record by reference (required)

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than or equal to 20000 is a warning, and a positive value below that indicates an index into the *item_status_array*, indicating the last element signaling an error condition.

Record type: **status_type** (Refer to appendix B.)

item_nums

32-bit signed integer array by reference (required)

An array of integers where each element is the number of an item (counter) to be enabled. If there are n item numbers being requested, element $n+1$ must be zero to indicate the end of the element list.

item_statuses

32-bit signed integer array by reference (required)

An array of n integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.

*mapping***32-bit signed integer by reference (required)**

This indicates the mapping for the object class counters requested by the user. The mapping currently in use is returned, which may be the one specified by the user or some other mapping chosen by an earlier MI user. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.

*ldev***32-bit signed integer by reference (optional)**

The logical device number (*ldev*) for which the counters are to be enabled. Specifying a value between -1 and -11 indicates the following:

- 1 Enable all configured logical devices in the system.
- 2 Enable all configured disks in the system.
- 3 Enable all configured tapes in the system.
- 4 Enable all configured terminals in the system.
- 5 Enable all configured printers in the system.
- 6 Enable all configured serial printers in the system.
- 7 Enable all configured spooler devices in the system.
- 8 Enable all configured data communication devices in the system.
- 9 Enable all configured DS terminals in the system.
- 10 Enable all configured DS printers in the system.
- 11 Enable all configured user devices in the system.

*user_id***32-bit signed integer by value (optional)**

The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSON, or the call fails.

Default: 0

Operation Notes

The specified logical device could be an LDEV number or a device type (for example, all disk devices or all tapes) for which counters are to be enabled. To monitor a specific set of logical devices, this procedure could be called repeatedly for those devices. This procedure should be called only once for every device type. If this procedure is called to monitor a single disk, no more disks can be monitored but tape devices can be monitored. If this procedure is called with a value of -1 in the *ldev* parameter, it is considered equivalent to calling this procedure ten times for each device type. In order to monitor a different device other than the ones that are currently monitored in a device type, all devices in that device type should be disabled.

Note

When all disks or all tapes are enabled, all devices of that type which were configured in the system at boot time are enabled. This is true even if those devices are not mounted.

MIPRCRGET

Retrieves processor counter values enabled by the caller for the desired processor(s).

Syntax

```

          REC          I32A          @64          I32A
MIPRCRGET ( overall_status, item_nums, buffer_ptr, item_statuses,
          I32          I32          I32          I32
          buffer_size, mapping, prcsr_id, num_of_prsrs,
          I32
          user_id );

```

Parameters*overall_status***record by reference (required)**

Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than or equal to 20000 is a warning, and a positive value below that indicates an index into the *item_status_array*, indicating the last element signaling an error condition.

Record type: *status_type* (Refer to appendix B.)

*item_nums***32-bit signed integer array by reference (required)**

An array of integers where each element is the number of an item to be returned. If there are *n* item numbers being requested, element *n*+1 must be zero to indicate the end of the element list.

*buffer_ptr***64-bit address by reference (required)**

A long pointer to a buffer that is sufficient to hold all items requested. The size of this buffer should be passed through the *buffer_size* parameter.

MIPRCRGET

<i>item_statuses</i>	32-bit signed integer array by reference (required) An array of n integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.
<i>buffer_size</i>	32-bit signed integer by value (required) The size of the buffer, in bytes.
<i>mapping</i>	32-bit signed integer by value (required) Indicates the mapping for the object class counters requested by the user. The mapping currently in use may be different from the one requested by the user. In such cases, a mapping is done from the one in use to the one requested by the user. Such mapping is possible only from a large set to a smaller set; that is, mapping is possible from a system map of large to a user map of small, and not vice versa. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.
<i>prcsr_id</i>	32-bit signed integer by reference (optional) The processor ID number for which counters are requested. Specify a <i>prcsr_id</i> of -1 or use the default value to request counters for all processors enabled by the user. Default: -1
<i>num_of_prcsrs</i>	32-bit signed integer by reference (optional) Indicates the number of processors for which information is returned in the buffer.
<i>user_id</i>	32-bit signed integer by value (optional) The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSON, or the call fails. Default: 0

Operation Notes

There is no explicit procedure to enable the counters specific to a processor. They are enabled implicitly when certain global counters are enabled. In order to enable processor counters you have to enable the item number 21164 along with other global counters. All processors are enabled but counter values specific to a processor Id could be retrieved through this procedure. The maximum number of processors that can be configured is 8 and the processor id's range from 0 to 7. Since there is no information available to indicate which processors are configured, this information could be dynamically retrieved by checking for nonzero values in certain counters such as gmc_launches (item number 27013).

Table 5-4. MIPCSRGET Item Information

Num	Name (Type) and Description
20040	Processor ID (I32) This returns the value of the Processor ID for which the counters are requested.

MIPROCGET

Retrieves process counter values enabled by the caller for the desired process(es).

Syntax

```
REC          I32A          @64          I32A
MIPROCGET ( overall_status, item_nums, buffer_ptr, item_statuses,
            I32          I32          REC          I32          I32
            buffer_size, mapping, pid, num_of_pids, miproc_ptr,
            I32
            user_id );
```

Parameters		
	<i>overall_status</i>	record by reference (required) Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than or equal to 20000 is a warning, and a positive value below that indicates an index into the <i>item_status_array</i> , indicating the last element signaling an error condition. Record type: <i>status_type</i> (Refer to appendix B.)
	<i>item_nums</i>	32-bit signed integer array by reference (required) An array of integers where each element is the number of an item to be returned. If there are <i>n</i> item numbers being requested, element <i>n</i> +1 must be zero to indicate the end of the element list.
	<i>buffer_ptr</i>	64-bit address by reference (required) A long pointer to a buffer that is sufficient to hold all items requested. The size of this buffer should be passed through the <i>buffer_size</i> parameter.

<i>item_statuses</i>	32-bit signed integer array by reference (required) An array of <i>n</i> integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.
<i>buffer_size</i>	32-bit signed integer by value (required) The size of the buffer, in bytes.
<i>mapping</i>	32-bit signed integer by value (required) Indicates the mapping for the object class counters requested by the user. The mapping currently in use may be different from the one requested by the user. In such cases, a mapping is done from the one in use to the one requested by the user. Such mapping is possible only from a large set to a smaller set; that is, mapping is possible from a system map of large to a user map of small, and not vice versa. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.
<i>pid</i>	record by reference (optional) The process ID (PID) number for which the counters are to be enabled. Specify a <i>pid</i> of -1 or use the default value to enable counters for all PIDs. Record_Type: <code>longint_type</code> (Refer to appendix B.) Default: -1
<i>num_of_pids</i>	32-bit signed integer by reference (optional) Indicates the number of PIDs for which information is returned in the buffer.
<i>miproc_ptr</i>	32-bit signed integer by reference (optional) This parameter is used as a fast key to retrieve information repeatedly for a specific process (PID). When initialized to zero, a value is passed through this parameter, which can be subsequently supplied for faster retrieval of information for this process.

MIPROCGET

user_id

32-bit signed integer by value (optional)

The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSON, or the call fails.

Default: 0

Operation Notes

Table 5-5. MIPROCGET Item Information

Num	Name (Type) and Description
20001	<p>PID (I64)</p> <p>This returns the value of the PID for which the counters are requested.</p>
20002	<p>PID State (I32)</p> <p>This returns the state of the PID. If the state of the process is dead, no more information for that PID is available to the user. The values returned are as follows:</p> <p>0: Alive 1: Dead</p>
20003	<p>User Name (rec)</p> <p>This returns the user name of the PID for which counter values are requested. It is of type <code>filename_type</code>.</p>
20004	<p>Program Name (rec)</p> <p>This returns the program name of the PID for which counter values are requested. It is of type <code>filename_type</code>.</p>
20005	<p>Start Time in Ticks Since 1970 (I64)</p> <p>This returns the start time (time of creation) of the PID in ticks since 1970.</p>
20006	<p>End Time in Ticks Since 1970 (I64)</p> <p>This returns the end time (time of death) of the PID in ticks since 1970.</p>

Table 5-5. MIPROCGET Item Information (continued)

Num	Name (Type) and Description
20007	<p>Start Time (rec)</p> <p>This returns the start time (time of creation) of the PID for which counter values are requested. It is of the following format:</p> <pre> Hour of Day = Start_Time(0:8) Minute of Hour = Start_Time(8:8) Second of Minute = Start_Time(16:8) Millisecond of Second = Start_Time(24:8) </pre>
20008	<p>End Time (rec)</p> <p>This returns the end time (time of death) of the PID for which counter values are requested. It is of the following format:</p> <pre> Hour of Day = End_Time(0:8) Minute of Hour = End_Time(8:8) Second of Minute = End_Time(16:8) Millisecond of Second = End_Time(24:8) </pre>
20009	<p>Start Date (rec)</p> <p>This returns the start date (date of creation) of the PID for which counter values are requested. It is of the following format:</p> <pre> Year after 1900 = Start_Date(0:7) Day of Year = Start_Date(7:9) Filler = Start_Date(16:16) </pre>
20010	<p>End Date (rec)</p> <p>This returns the end date (date of death) of the PID for which counter values are requested. It is of the following format:</p> <pre> Year after 1900 = End_Date(0:7) Day of Year = End_Date(7:9) Filler = End_Date(16:16) </pre>
20011	<p>Job/Session Number (rec)</p> <p>Returns the job/session number for the PID in the form <code>jsnum_type</code>.</p>
20012	<p>Input Device (I32)</p> <p>Returns the LDEV associated with <code>\$STDIN</code> for the PID.</p>

MIPROCOFF

Disables the process-related counters enabled by the user.

Syntax

REC	REC	I32
MIPROCOFF (<i>overall_status</i> , <i>pid</i> , <i>user_id</i>)		

Parameters		
	<i>overall_status</i>	record by reference (required) Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error, and a positive value greater than or equal to 20000 indicates a warning. Record type: status_type (Refer to appendix B.)
	<i>pid</i>	record by reference (optional) The Process ID (PID) number for which the counters are to be disabled. Specifying a value of -1 or using the default value would disable all processes enabled by the user. Record type: longint_type (Refer to appendix B.) Default: -1
	<i>user_id</i>	32-bit signed integer by value (optional) The user ID assigned to a vendor at the time of purchase of the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using AIFACCESSION, or the call fails. Default: 0

Operation Notes

When MIPROCOFF is called, all counters enabled for a specified process are disabled. The user has the option to disable only for certain processes and still to continue to collect information for other processes. When all processes in the system are enabled and a specific process is disabled, the “all process” capability is nullified; that is, only processes in the system that existed at the time of disable (not including the particular process disabled) are monitored. No more new processes are monitored.

MIPROCON

Allows a user to turn on counters either for a single process or for all processes on the system.

Syntax

```

REC          I32A          I32A          I32          REC
MIPROCON ( overall_status, item_nums, item_statuses, mapping, pid,
           I32
           user_id );

```

Parameters	<i>overall_status</i>	record by reference (required)
		<p>Holds the overall status of the call. A negative value indicates an error in the overall call, not specific to any particular item; a 0 indicates no error. A positive value greater than and equal to 20000 indicates a warning, and a positive value below that indicates an index into the <i>item_status_array</i>, indicating the last element signaling an error condition.</p> <p>Record type: <code>status_type</code> (Refer to appendix B.)</p>
	<i>item_nums</i>	<p>32-bit signed integer array by reference (required)</p> <p>An array of integers where each element is the number of an item (counter) to be enabled. If there are <i>n</i> item numbers being requested, element <i>n</i>+1 must be zero to indicate the end of the element list.</p>
	<i>item_statuses</i>	<p>32-bit signed integer array by reference (required)</p> <p>An array of <i>n</i> integers where each element returns the status for the corresponding item. A negative value indicates an error condition, a zero indicates normal execution, and a positive value indicates a warning.</p>

<i>mapping</i>	<p>32-bit signed integer by reference (required)</p> <p>Indicates the mapping for the object class counters requested by the user. The mapping currently in use is returned. The mapping may be the one specified by the user or a mapping specified by an earlier MI user. The corresponding values for large, small, and standard map are 0, 1, and 2 respectively. All other values are considered illegal.</p>
<i>pid</i>	<p>record by value (optional)</p> <p>The process ID (PID) number for which the counters are to be enabled. Specify a <i>pid</i> of -1 or use the default value to enable counters for all PIDs.</p> <p>Record_Type: <code>longint_type</code> (Refer to appendix B.)</p> <p>Default: -1</p>
<i>user_id</i>	<p>32-bit signed integer by value (optional)</p> <p>A user ID assigned to a vendor at the time of purchase the Architected Interface Facility: Measurement Interface product. If this is supplied, it is used to confirm that the caller has the AIF product. If it is not passed, the caller should have switched access on using <code>AIFACCESSION</code>, or the call fails.</p> <p>Default : 0</p>

Operation Notes

To monitor performance on a single process or a set of processes, process-specific counters can be enabled using `MIPROCON`. A single call to `MIPROCON` allows a user to turn on counters either for a single process or for all processes on the system. In order to enable a specific set of processes, `MIPROCON` has to be called repeatedly for each of the processes.

If `MIPROCON` is called multiple times, only the same set of counters should be turned on for various processes. If a different set of counters are required, the current set of counters should be disabled before calling `MIPROCON`. When a value of -1 is passed through the *pid* parameter to monitor all processes in the system, included are the existing processes and all new processes born after this call.

AIF Status Messages

Table A-1. Errors and Warnings Returned

AIFErr	1) Message 2) Cause 3) Action
-1	<ol style="list-style-type: none"> 1. Read probe failed. 2. Caller does not have read access to a virtual address. 3. Check for uninitialized pointers.
-2	<ol style="list-style-type: none"> 1. Write probe failed. 2. Caller does not have write access to a virtual address. 3. Check for uninitialized pointers.
-3	<ol style="list-style-type: none"> 1. Read nonscalar probe failed. 2. Caller does not have read access to a series of pages in VSM. 3. Check for uninitialized pointers and counts.
-4	<ol style="list-style-type: none"> 1. Write nonscalar probe failed. 2. Caller does not have write access to a series of pages in VSM. 3. Check for uninitialized pointers and counts.
-5	<ol style="list-style-type: none"> 1. Bad pointer was encountered. 2. The address is uninitialized. 3. Check for uninitialized pointers.
-6	<ol style="list-style-type: none"> 1. Badly aligned pointer was encountered. 2. The pointer is not correctly aligned. 3. Check for uninitialized pointers and alignment requirements.
-7	<ol style="list-style-type: none"> 1. A value mismatch was encountered. 2. The value passed in version array is not the same as an integer value. 3. Call AIFGET to get correct value.
-8	<ol style="list-style-type: none"> 1. Array overflow. 2. The dynamic length array passed in was too small to hold all values. 3. Call AIFSCGET to get upper bounds on array sizes required.
-20	<ol style="list-style-type: none"> 1. Verification arrays wrongly specified. 2. Verification arrays to PUT were wrongly specified. 3. Pass all three or none at all.
-21	<ol style="list-style-type: none"> 1. Bad overall status. 2. The overall status was inaccessible for write access. 3. Check for uninitialized pointers.
-22	<ol style="list-style-type: none"> 1. Bad item status. 2. The item status was inaccessible for write access. 3. Check for uninitialized pointers.

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-23	<ol style="list-style-type: none"> 1. Bad verification item status. 2. The verification item status was inaccessible for write access. 3. Check for uninitialized pointers.
-24	<ol style="list-style-type: none"> 1. Verification failed. 2. The verification for PUT failed. 3. Check the version item statuses for more information.
-25	<ol style="list-style-type: none"> 1. Incorrect user capability. 2. The AIF procedure called is not accessible with the specified user ID. 3. Purchase the AIF product component referenced.
-26	<ol style="list-style-type: none"> 1. Non-existent MI User ID. 2. The user ID specified does not exist. 3. Use user ID distributed when Architected Interface Facility: Measurement Interface was purchased.
-27	<ol style="list-style-type: none"> 1. Invalid search key. 2. The AIFSWSWIDEGET search key is no longer valid. 3. Restart AIFSWSWIDEGET calls.
-28	<ol style="list-style-type: none"> 1. Invalid JSNum. 2. The job/session specified does not exist. 3. Verify if the JSNum exists.
-29	<ol style="list-style-type: none"> 1. PID/PIN mismatch encountered. 2. The PID process has died, and a new process with same PIN was born. 3. Check the PID and the PIN.
-30	<ol style="list-style-type: none"> 1. Process is dead. 2. No process with this PIN exists on the system. 3. Check the PIN.
-31	<ol style="list-style-type: none"> 1. The process is not of type user or son or CI. 2. Attempt to PUT to a process of type not user or son. 3. Check the process type and the PIN/PID.
-32	<ol style="list-style-type: none"> 1. Invalid accounting name. 2. AIFACCTGET/PUT could not find the specified account name. 3. Verify that the user, group, and account specified exist.
-33	<ol style="list-style-type: none"> 1. Invalid fnum PID combination. 2. The PID process does not have a file open with this fnum. 3. Check the PID/fnum combination.
-34	<ol style="list-style-type: none"> 1. A device file was encountered where it is not supposed to be. 2. Attempt to PUT to a file of device type. 3. Check the file type and the fnum/PID combination.
-35	<ol style="list-style-type: none"> 1. The UFID does not correspond to the file specified. 2. The fnum was closed and a new file was opened with same fnum. 3. Check the list of open files using AIFPROCGET.

A-2 AIF Status Messages

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-36	<ol style="list-style-type: none"> 1. Not a user file. 2. Attempt to PUT to a file with designator not user. 3. Check the file designator.
-37	<ol style="list-style-type: none"> 1. A directory object was encountered. 2. Attempt to PUT to a file that is actually a Directory object. 3. Check for Dir Obj. in AIFFILELGET
-38	<ol style="list-style-type: none"> 1. Parameter 1 was badly aligned. 2. Parameter 1 was badly aligned. 3. Check for uninitialized pointers.
-39	<ol style="list-style-type: none"> 1. Parameter 2 was badly aligned. 2. Parameter 2 was badly aligned. 3. Check for uninitialized pointers.
-40	<ol style="list-style-type: none"> 1. Parameter 3 was badly aligned. 2. Parameter 3 was badly aligned. 3. Check for uninitialized pointers.
-41	<ol style="list-style-type: none"> 1. Parameter 4 was badly aligned. 2. Parameter 4 was badly aligned. 3. Check for uninitialized pointers.
-42	<ol style="list-style-type: none"> 1. Invalid UFID. 2. The UFID parameter specified does not exist. 3. Verify the UFID used.
-43	<ol style="list-style-type: none"> 1. Invalid file name. 2. The file name specified does not exist. 3. Verify if the file name exists.
-45	<ol style="list-style-type: none"> 1. Return array1 write probe failed. 2. User does not have write access to the array passed in. 3. Check for uninitialized pointers and num_array_entry.
-46	<ol style="list-style-type: none"> 1. Return array2 write probe failed. 2. User does not have write access to the array passed in. 3. Check for uninitialized pointers and num_array_entry.
-47	<ol style="list-style-type: none"> 1. Invalid AIF key. 2. AIFSYSWIDEGET did not recognize the aif_area key. 3. Try 1000, 2000, 5000, 6000, 8000, or 11000.
-48	<ol style="list-style-type: none"> 1. Creation of sharable object failed. 2. Call to AIFGLOBACQ was unsuccessful. 3. Possibly out of transient disk space.
-49	<ol style="list-style-type: none"> 1. Release of sharable object failed. 2. Call to AIFGLOBREL was unsuccessful. 3. Verify the object pointer is valid.

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-50	<ol style="list-style-type: none"> 1. Missing criteria arrays. 2. AIFSYSWIDEGET AIF key specified requires criteria arrays. 3. Use itemnum_array, item_array, item_status_array parameters.
-55	<ol style="list-style-type: none"> 1. AIFCLOSE failed. 2. Either a bad file_number was specified, another file with the same name already exists, an illegal disposition (5, 6, 7) or any outstanding write I/Os may have failed. 3. Use FCHECK to determine the reason that AIFCLOSE failed.
-56	<ol style="list-style-type: none"> 1. The address passed for the verification item number array is not accessible to the caller. 2. The address passed is not accessible to the caller. 3. Pass only addresses in accessible spaces.
-57	<ol style="list-style-type: none"> 1. The address passed for the verification items array is not accessible to the caller. 2. The address passed is not accessible to the caller. 3. Pass only addresses in accessible spaces.
-58	<ol style="list-style-type: none"> 1. The address is not properly aligned for the verification item number to be accessible. 2. The address is not properly aligned for the data type to be accessed. 3. Pass only variable that has the proper data alignment.
-59	<ol style="list-style-type: none"> 1. The address is not properly aligned for the verification items to be accessed. 2. The address is not properly aligned for the data type to be accessed. 3. Pass only variable that has the proper data alignment.
-60	<ol style="list-style-type: none"> 1. The address is not properly aligned for the verification item statuses. 2. The address is not properly aligned for the data type to be accessed 3. Pass only variable that has the proper data alignment
-100	<ol style="list-style-type: none"> 1. Internal error. Can't find process entry for caller. 2. Internal inconsistency. 3. Take a system dump. Contact your Hewlett-Packard support representative.
-101	<ol style="list-style-type: none"> 1. Unsupported option. 2. Port manage access was requested, but is not supported. 3. Do not attempt to open a port for port manager access.
-102	<ol style="list-style-type: none"> 1. Too many receive opens on the port. 2. The maximum number of receive opens have already been done. 3. Check the logic of your application. The maximum is very large.

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-103	<ol style="list-style-type: none"> 1. Too may opens for manage access on the port. 2. The maximum number of manage opens have already been done. 3. Check the logic of your application.
-104	<ol style="list-style-type: none"> 1. Too many opens for send access on the port. 2. The maximum number of send opens have already been done. 3. Check your application. The maximum is very large.
-105	<ol style="list-style-type: none"> 1. Invalid ACCESSMODE specified. 2. The access mode code is not one of the allowed values. 3. Check the logic of your application.
-106	<ol style="list-style-type: none"> 1. Message length is negative or greater than maximum. 2. The specified message length is not valid. 3. Check the logic of your application.
-107	<ol style="list-style-type: none"> 1. Specified port ID is not valid. 2. The port is either not open, or is invalid. 3. Check the logic of your application.
-108	<ol style="list-style-type: none"> 1. Attempted to send a message on port not opened for send access. 2. The calling process does not have the port open for send access. 3. Check the logic of your application.
-109	<ol style="list-style-type: none"> 1. Attempted to receive a message from port not open for receive access. 2. The calling process does not have the port open for receive access. 3. Check the logic of your application.
-110	<ol style="list-style-type: none"> 1. Attempted to manage a port not open for manage access. 2. The calling process is not the port manager. 3. Check the logic of your application.
-111	<ol style="list-style-type: none"> 1. A timeout occurred. 2. The specified number of seconds has passed. 3. Verify that the timeout value specified is sufficient.
-112	<ol style="list-style-type: none"> 1. No ports open for receive access, multiport receive failed. 2. The calling process has no ports open for receive access. 3. Check the logic of your application.
-113	<ol style="list-style-type: none"> 1. Attempt to open port for same access multiple times. 2. Process attempted to open same port for same access multiple times. 3. Check the logic of your application.
-114	<ol style="list-style-type: none"> 1. Unsupported procedure. 2. A procedure that is not yet supported was called. 3. Check the logic of your application.
-115	<ol style="list-style-type: none"> 1. No zero element terminator was found in the itemnums array. 2. No terminator was found in the itemnums array. 3. Check the logic of your application.

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-116	<ol style="list-style-type: none"> 1. Invalid password. 2. The named port exists, but the password supplied does not match. 3. Check the logic of your application.
-117	<ol style="list-style-type: none"> 1. Internal error. 2. The port does not exist. 3. Take a dump. Contact your Hewlett-Packard support representative.
-118	<ol style="list-style-type: none"> 1. Internal error. 2. Could not obtain ownership for any Ports on list. 3. Take a dump. Contact your Hewlett-Packard support representative.
-119	<ol style="list-style-type: none"> 1. Internal error. 2. Table management error when obtaining open table entry. 3. Take a dump. Contact your Hewlett-Packard support representative.
-120	<ol style="list-style-type: none"> 1. Internal error. 2. Problem linking receive open to the port. 3. Take a dump. Contact your Hewlett-Packard support representative.
-121	<ol style="list-style-type: none"> 1. Internal error. 2. Problem creating reply port for receive open. 3. Take a dump. Contact your Hewlett-Packard support representative.
-122	<ol style="list-style-type: none"> 1. Internal error. 2. Problem linking send open to the port. 3. Take a dump. Contact your Hewlett-Packard support representative.
-123	<ol style="list-style-type: none"> 1. Internal error. 2. Problem creating the reply port for a send open. 3. Take a dump. Contact Your Hewlett-Packard support representative.
-124	<ol style="list-style-type: none"> 1. Internal error. 2. Problem linking manage open to the port. 3. Take a dump. Contact Your Hewlett-Packard support representative.
-125	<ol style="list-style-type: none"> 1. Itemnums, items, and itemstatus not specified together. 2. Must pass item option arrays as a triple. All or none. 3. Check the logic of your application.
-126	<ol style="list-style-type: none"> 1. Must complete two-part receive before receive from another port. 2. Receive from second port before doing second part of two part receive. 3. Check the logic of your application.
-127	<ol style="list-style-type: none"> 1. Must request message return on second receive of two-part receive. 2. The second receive of a two-part receive did not request message. 3. Check the logic of your application.
-128	<ol style="list-style-type: none"> 1. The message length specified on send was larger than the maximum length specified when the Port was created. 2. Message can not be larger than the specified size. 3. Check the logic of your application.

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-129	<ol style="list-style-type: none"> 1. A NOWAIT receive was done while there was no message ready. 2. A NOWAIT receive was done while there was no message ready. 3. Check the logic of your application.
-3014	<ol style="list-style-type: none"> 1. Invalid value passed in AIFTIME. 2. A negative or otherwise invalid value was passed in AIFTIME. 3. Pass a positive value to AIFTIME.
-20001	<ol style="list-style-type: none"> 1. Counters were already enabled by user. 2. The user had previously enabled a set of counters or has reached the limit in the number of ON calls. 3. Check for limitations in ON calls.
-20002	<ol style="list-style-type: none"> 1. Cannot add additional MI users. 2. The MI user limit is reached and additional users cannot be enabled. 3. Wait for other MI users to be done.
-20003	<ol style="list-style-type: none"> 1. Cannot enable additional LDEVs. 2. No space in MI tables. 3. Reduce number of LDEVs being monitored.
-20004	<ol style="list-style-type: none"> 1. Cannot enable additional counters. 2. The upper limit is reached on the number of counters enabled. 3. Reduce number of counters being monitored on the system.
-20005	<ol style="list-style-type: none"> 1. The specified PID is not present. 2. The PID does not exist, has died, or has not been enabled by the user. 3. Don't request information on non-existent processes.
-20006	<ol style="list-style-type: none"> 1. An invalid item number was specified in itemnums array. 2. A nonzero, invalid number was passed. 3. Pass an appropriate item number in itemnum array.
-20007	<ol style="list-style-type: none"> 1. Invalid value was passed for object class mapping. 2. The value passed is not a valid object class mapping. 3. Pass an appropriate object class mapping value.
-20008	<ol style="list-style-type: none"> 1. Buffer size is insufficient. 2. The size of the buffer passed is insufficient to hold return values. 3. Pass a buffer size large enough to hold all return values.
-20009	<ol style="list-style-type: none"> 1. Process data is not available. 2. All processes requested by user have died or were not enabled. 3. Don't request information on non-existent processes.
-20010	<ol style="list-style-type: none"> 1. Cannot enable the LDEVs requested. 2. The device limit has been exceeded. 3. Reduce number of LDEVs being monitored.
-20011	<ol style="list-style-type: none"> 1. Requested counter has been deleted. 2. The counter requested is no longer valid. 3. Do not request the deleted counter.

Table A-1. Errors and Warnings Returned (continued)

AIFErr	1) Message 2) Cause 3) Action
-20012	<ol style="list-style-type: none"> 1. No valid items passed in itemnums array. 2. All items requested are not valid. 3. Check the manual and pass only valid items in itemnums array.
-20013	<ol style="list-style-type: none"> 1. MI user is not enabled. 2. The user has not enabled before trying to access counters. 3. Enable counters prior to trying to access them.
-20014	<ol style="list-style-type: none"> 1. Illegal mapping conversion requested in GET AIFs 2. The requested mapping conversion is not legal. 3. Request a legal mapping conversion.
-20015	<ol style="list-style-type: none"> 1. Insufficient buffer space was provided to load all counters. 2. The number of counters requested exceeds the buffer space. 3. Provide a buffer space large enough to hold all counters requested.
-20016	<ol style="list-style-type: none"> 1. Invalid LDEV or negative number specified in ldev parameter. 2. The negative number representing a device type or LDEV is invalid. 3. Specify a valid negative number or LDEV.
-20017	<ol style="list-style-type: none"> 1. Could not enable all devices or a type of device requested. 2. All devices or a type of device could not be enabled. 3. Reduce the number of counters being monitored on the system.
-20018	<ol style="list-style-type: none"> 1. The buffer pointer passed is invalid. 2. The address is uninitialized. 3. Check for uninitialized pointers.
-20019	<ol style="list-style-type: none"> 1. The buffer pointer passed is badly aligned. 2. The pointer is not correctly aligned. 3. Check for uninitialized pointers and alignment requirements.
-20020	<ol style="list-style-type: none"> 1. The process pointer miprocc_ptr is bad. 2. The address is uninitialized. 3. Check for uninitialized pointers.
-20021	<ol style="list-style-type: none"> 1. The process pointer miprocc_ptr is badly aligned. 2. The pointer is not correctly aligned. 3. Check for uninitialized pointers and alignment requirements.
20001	<ol style="list-style-type: none"> 1. Not all process data was reported to user. 2. Some processes were not tracked due to insufficient space. 3. Reduce the number of counters being monitored on the system.
20002	<ol style="list-style-type: none"> 1. Object class mapping requested was not given. 2. A different mapping was previously selected by another MI user. 3. Use convert map option if possible, the selected mapping or halt.

AIF Data Structures

```
bit1          = 0 .. 1;

bit2          = 0 .. 3;

bit8          = 0 .. 255;

bit14         = 0 .. 16383;

bit16         = 0 .. 65535;

bit31        = 0 .. 2147483647;
```

```
clock_type =

  crunched record
  case bit32 of
    1: ( hour      : bit8;      (0.0 @ 1.0)
        min       : bit8;      (1.0 @ 1.0)
        sec       : bit8;      (2.0 @ 1.0)
        ten_sec   : bit8;      (3.0 @ 1.0)
    2: ( clock_funct : bit32;   (0.0 @ 4.0)
  end;

Record size: 4 bytes
Alignment: 4 bytes
```

```
datestr_type =
  record
    month_str      : packed array[1..3] of char;  (0.0 @ 3.0)
    day_of_week    : packed array[1..3] of char;  (3.0 @ 3.0)
  end;
```

Record size: 6 bytes
Alignment: 1 byte

```
date_type =
  record
    year          : integer;  (0.0 @ 4.0)
    month         : integer;  (4.0 @ 4.0)
    day_of_month  : integer;  (8.0 @ 4.0)
  end;
```

Record size: 12 bytes
Alignment: 4 bytes

```
device_name_type      =
    packed array [ 1 .. 18 ] of char;          ( 0.0 @ 18.0 )
```

Array size : 18 bytes
Element size : 1 byte
Alignment : 1 byte

```
directory_name_type =
    record
      user      : packed array [ 1 .. 16 ] of char ;  ( 0.0 @ 16.0 )
      group     : packed array [ 1 .. 16 ] of char ;  ( 16.0 @ 16.0 )
      account   : packed array [ 1 .. 16 ] of char ;  ( 32.0 @ 16.0 )
    end ;
```

Record size : 48 bytes
Alignment : 1 byte

The values for the user, group, and account must be uppercase.

```

dstsrec_type =

    record
        cnt      : integer;                ( 0.0 @ 4.0 )
        dstinfo :
            array [ 1 .. x ] of
                record
                    dstno  : integer;        ( 0.0 @ 4.0 )
                    dstva  : anyptr;        ( 4.0 @ 8.0 )
                end ;
            end ;
    end ;

```

Record size : $12x + 4$ bytes

Alignment : 4 bytes

Here x is a user-defined size.

```

filename_type =

    record
        filename : packed array[1..16] of char; ( 0.0 @ 16.0 )
        group    : packed array[1..16] of char; ( 16.0 @ 16.0 )
        account  : packed array[1..16] of char; ( 32.0 @ 16.0 )
    end ;

```

Record size : 48 bytes

Alignment : 1 byte

All three arrays are always returned padded with blanks on the right.
 Most interfaces accept them, when a file name is input, to be flushed
 with blanks on the right.

```
fnumpid_type =  
  
    record  
        fnum      : integer;           ( 0.0 @ 4.0 )  
        pid       : longint_type;     ( 4.0 @ 8.0 )  
    end ;  
  
Record size : 12 bytes  
Alignment   : 4 bytes
```

```
item_array_type      =  
  
    array [ 1 .. x ] of globalanyptr;  
  
Array size   : 8x bytes  
Element size : 8 bytes  
Alignment    : 4 bytes  
  
This is user defined-type. Here x could be any number.
```

```
item_status_array_type =  
  
    array [ 1 .. x ] of status_type;  
  
Array size   : 4x bytes  
Element size : 4 bytes  
Alignment    : 4 bytes  
  
This is a user-defined type. Here x could be any number.
```



```
itemnum_array_type =  
  
    array [ 1 .. x ] of integer
```

```
Array size   : 4x bytes  
Element size : 4 bytes  
Alignment    : 4 bytes
```

This is a user-defined type. Here x could be any number.

```
jskey_type      =  
  
    integer;                ( 0.0 @ 4.0 )  
  
Alignment       : 4 bytes
```

```
jsdev_type =  
  
    record  
        device_class : boolean;      ( 0.0 @ 1.0 )  
        output_device : integer;     ( 4.0 @ 4.0 )  
    end ;  
  
Record size : 8 bytes  
Alignment   : 4 bytes
```

```

jsnum_type      =

    packed record
        js_type  : bit2;           ( 0.0 @ 0.2 )
        js_num   : bit14;          ( 0.2 @ 1.6 )
        js_ext   : shortint;       ( 2.0 @ 2.0 )
    end ;

```

Record size : 4 bytes

Alignment : 1 bit

The js_type field can have a value of 1 or 2, where a 1 indicates a session and a 2 indicates a job. The js_num field can be any value from 1-16383.

```

logon_desc_type =

    record
        job_name      : pac16;      ( 0.0 @ 16.0 )
        acct_name     : pac16;      ( 16.0 @ 16.0 )
        acct_pass     : pac16;      ( 32.0 @ 16.0 )
        user_name     : pac16;      ( 48.0 @ 16.0 )
        user_pass     : pac16;      ( 64.0 @ 16.0 )
        group_name    : pac16;      ( 80.0 @ 16.0 )
        group_pass    : pac16;      ( 96.0 @ 16.0 )
    end ;

```

Record size : 112 bytes

Alignment : 1 byte

```

longint_type =

    record
        left   : integer;          ( 0.0 @ 4.0 )
        right  : integer;          ( 4.0 @ 4.0 )
    end ;

```

Record size : 8 bytes

Alignment : 4 bytes

```
message_buffer_type =  
    packed array [ 1 .. x ] of char;
```

```
Array size   : x bytes  
Element size : 1 byte  
Alignment    : 1 byte
```

This is a user defined type. Here *x* could be any number ≤ 32767 .

```
pac8          =  
    packed array [ 1 .. 8 ] of char;          ( 0.0 @ 8.0)
```

```
Array size   : 8 bytes  
Element size : 1 byte  
Alignment    : 1 byte
```

```
pac16         =  
    packed array [ 1 .. 16 ] of char;        ( 0.0 @ 16.0)
```

```
Array size   : 16 bytes  
Element size : 1 byte  
Alignment    : 1 byte
```

```
pac18         =  
    packed array [ 1 .. 18 ] of char;        ( 0.0 @ 18.0)
```

```
Array size   : 18 bytes  
Element size : 1 byte  
Alignment    : 1 byte
```

```
pac32          =  
  
    packed array [ 1 .. 32 ] of char;          ( 0.0 @ 32.0)  
  
Array size    : 32 bytes  
Element size  : 1 byte  
Alignment     : 1 byte
```

```
pac34          =  
  
    packed array [ 1 .. 34 ] of char;          ( 0.0 @ 34.0)  
  
Array size    : 34 bytes  
Element size  : 1 byte  
Alignment     : 1 byte
```

```
pac256         =  
  
    packed array [ 1 .. 256 ] of char;        ( 0.0 @ 256.0)  
  
Array size    : 256 bytes  
Element size  : 1 byte  
Alignment     : 1 byte
```

```
pid_type =  
  
    record  
        left    : integer;          ( 0.0 @ 4.0 )  
        right   : integer;          ( 4.0 @ 4.0 )  
    end ;  
  
Record size   : 8 bytes  
Alignment     : 4 bytes
```

```

recfnumpid_type =

    record
        count      : integer;                ( 0.0 @ 4.0 )
        fnumpid    : array[1..x] of fnumpid_type ( 4. 0 @ 12x.0)
    end ;

```

Record size : $12x + 4$ bytes

Alignment : 4 bytes

Here x is a user defined size.

```

search_key_type      =

    record
        case integer of
            1: ( key_js_num  : integer );          ( 0.0 @ 4.0)
            2: ( key_pid     : longint_type );     ( 0.0 @ 8.0)
            3: ( key_ufile   : ufile_type );       ( 0.0 @ 20.0)
            4: ( key_fname   : filename_type );    ( 0.0 @ 48.0)
            5: ( key_dname   : directory_name_type ); ( 0.0 @ 48.0)
            6: ( key_sfnum   : integer );          ( 0.0 @ 4.0)
            7: ( key_portid  : integer );          ( 0.0 @ 4.0)
            8: ( key_portnm  : pac16 );           ( 0.0 @ 16.0)
            9: ( key_plfd    : localanyptr );     ( 0.0 @ 4.0)
            10: ( key_js_ind : integer );          ( 0.0 @ 4.0)
            11: ( key_pid_ind: integer );          ( 0.0 @ 4.0)
        end ;

```

Record size : 48 bytes

Alignment : 4 bytes

```

sel_eq_type      =

    record
        stringlen: integer ;                ( 0.0 @ 4.0 )
        str : packed array [ 1 .. 280 ] of char ; ( 4.0 @ 280.0)
        housekeep: bit8;                    ( 280.0 @ 1.0)
    end ;

Record size : 288 bytes
Alignment   : 4 bytes

```

```

spf_id_type      =

    packed record
        case boolean of
            true: ( id_number  : bit31;      ( 0.0 @ 3.7 )
                  i_or_o_flag: bit1);      ( 3.7 @ 0.1 )
            false: ( all: integer);        ( 0.0 @ 4.0 )
        end ;

Record size : 4 bytes
Alignment   : 4 bytes

```

```

status_type      =

    record
        case boolean of
            true  : ( all: integer );        ( 0.0 @ 4.0 )
            false : ( info  : shortint;     ( 0.0 @ 2.0 )
                    subsys: shortint );    ( 2.0 @ 2.0 )
        end ;

Record size : 4 bytes
Alignment   : 4 bytes

```

```
ufid_type      =  
    record  
        ufid : packed array [ 1 .. 20 ] of char;    ( 0.0 @ 14.0)  
    end ;
```

```
    Array size: 20 bytes  
    Element size: 1 byte  
    Alignment : 4 bytes
```


Object Class Information

This appendix provides information relating to object class counters. All object class counters are returned as an array of integers. The length of the array depends on the mapping chosen and is given below.

Map	Length of Array
Standard	10
Small	33
Large	720

Each element in the array relates to a collection of objects in the system, collectively known as an object class. The collection of objects, hence the object class, depends on the mapping chosen. This appendix gives the object class information corresponding to each index in the array for various mappings. All arrays are assumed to start with an index of 0.

Table C-1. Object Class Information for Standard Map

Array Index	Object Class Name and Description
0	objcl_transient_data Unspecified transient data object
1	objcl_permanent_data Unspecified permanent data object
2	objcl_nm_stack NM stack
3	objcl_cm_stack CM stack
4	objcl_nm_code NM code
5	objcl_cm_code CM code (program file)
6	objcl_cm_data CM data segment
7	objcl_file_object File object
8	objcl_nm_sys_lib NM system library
9	objcl_cm_sys_lib CM system library

Table C-2. Object Class Information for Small Map

Array Index	Object Class Name and Description
0	objcl_transient_data Unspecified transient data object
1	objcl_permanent_data Unspecified permanent data object
2	objcl_nm_stack NM stack
3	objcl_cm_stack CM stack
4	objcl_nm_code NM code
5	objcl_cm_code CM code (program file)
6	objcl_cm_data CM data segment
7	objcl_file_object File object
8	objcl_nm_sys_lib NM system library
9	objcl_cm_sys_lib CM system library
10	objcl_virtual_space All of VSMs objects
11	objcl_system_globals All of system globals
12	objcl_sec_storage All secondary storage mgr
13	objcl_user_interface_js All of user interface
14	objcl_file_label_table All of file label management

Table C-2. Object Class Information for Small Map (continued)

Array Index	Object Class Name and Description
15	objcl_stg_volumn_file_type All of storage management, volume management, and file system type managers
16	objcl_ports All of ports known global known process ports
17	objcl_loader All of loader
18	objcl_process_manager All of process management
19	objcl_unknown All of memory management
20	objcl_hpsql All of HP SQL
21	objcl_measurement_interface All of measurement interface
22	objcl_hlio_lowlevel_io All of HLIO and low-level I/O
23	objcl_data_comm All of datacomm
24	objcl_turbo_buffer All TurboIMAGE map table and Turbo IMAGE buffers
25	objcl_xm All of transaction management
26	objcl_cm_mpe_compatibility All of compatibility mode objects, compatibility mode object management, and general compatibility mode DSTs
27	objcl_ksam_files All of KSAM files
28	objcl_cm_files All of CM files

Table C-2. Object Class Information for Small Map (continued)

Array Index	Object Class Name and Description
29	objcl_directory_files All of directory files
30	objcl_nm_subsys_lib_code NM subsystem library
31	objcl_Turbo_files All Turbo files
32	objcl_xm_files All XM files

Table C-3. Object Class Information for Large Map

Array Index	Object Class Name and Description
0	objcl_transient_data Unspecified transient data object
1	objcl_permanent_data Unspecified permanent data object
2	objcl_nm_stack NM stack
3	objcl_cm_stack CM stack
4	objcl_nm_code NM code
5	objcl_cm_code CM code (program file)
6	objcl_cm_data CM data segment
7	objcl_file_object File object
8	objcl_nm_sys_lib NM system library
9	objcl_cm_sys_lib CM system library
10	objcl_shared_page Unidentified shared object page
11	objcl_vsm_control_obj VSM control object
12	objcl_vs_b_tree_table Virtual space B-tree table
13	objcl_vpn_cache_table VPN resident translation cache
14	objcl_vpn_ar_cache_table VPN cache with AR and PIDs

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
15	objcl_vs_od_table Virtual space block descriptor table
16	objcl_ext_b_tree_table Extent B-tree Table
17	objcl_ext_ar_b_tree_table Extent B-tree table with AR and PIDs
18	objcl_shr_page_desc_table Shared page descriptor table
19	objcl_vs_domain_desc_table VS domain descriptor table
20	objcl_vs_recovery_table VS recovery stack table
21	objcl_vs_od_gu_fd_table VS-OD/global unique file descriptor table
22	objcl_indirect_loc_table Indirect locality table
23	objcl_dis_exp_vs_od_table Disabled expandable VS_ODs
24	objcl_system_globals Virtual system global object
25	objcl_real_globals Real system global object
26	objcl_ssm_map Secondary space bit maps
27	objcl_ssm_partial_page_map SSM partial page map
28	objcl_ssm_dir Secondary space directory
29	objcl_ci_tmp

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
30	objcl_reply_information_table System reply information table
31	objcl_variable_table CI variable table
32	objcl_udc_dir UDC directory
33	objcl_while_stack CI while stack
34	objcl_process_ccb CI process command control block
35	objcl_break_ccb CI break command control block
36	objcl_dwf Object to keep track of dump-worthy executable files, used by the loader and DAT/dump. CI RIT break command control block
37	objcl_programmatic_ccb CI programmatic command control block
38	objcl_jobmain_port Job main port
39	objcl_sessionmain_port Session main port
40	objcl_jobqueue_port Job queue port
41	objcl_jsmain_port Job/Session main port
42	objcl_aif Architected Interfaces KSO
43	objcl_sysmain_term_buf Sysmain terminal buffers
44	objcl_lab_tbl File label table

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
45	objcl_mvmt Mounted volume table
46	objcl_mvlt Mounted volume LDEV table
47	objcl_vsvidt Mounted volume set volume ID table
48	objcl_mvst Mounted volume set table
49	objcl_vsit Volume set information table
50	objcl_sm_globals Storage management globals
51	objcl_nlio_buf NLIO buffer
52	objcl_nm_debug NM debug break point table
53	objcl_cm_debug_globals CM Debug globals
54	objcl_global_port_object Global port object
55	objcl_non_res_port_desc_tbl Non-resident port descriptor table
56	objcl_res_port_desc_tbl Resident port descriptor table
57	objcl_port_state_table Port descriptor array
58	objcl_wait_queue_table Global wait queue table
59	objcl_port_directory Global port name directory

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
60	objcl_port_control_block Port control block
61	objcl_message_pool Message pool for a port
62	objcl_fio_state_tbl File IO state table
63	objcl_port_and_pool Object containing port and pool
64	objcl_local_desc_table Process local port descriptor table
65	objcl_port_freeze_tbl Port freeze descriptor table
66	objcl_port_std_message_pool Message pool for process std ports
67	objcl_port_delay_message_pool Message pool for delayed requests
68	objcl_shr_sem_table Shared semaphore table
69	objcl_ldr_globals_table Loader globals table
70	objcl_ldr_process_xrt Process XRT object
71	objcl_ldr_system_xrt System XRT
72	objcl_ldr_lft Loader file table object
73	objcl_ldr_pfl_table Process file list
74	objcl_ldr_unsat_table Unsat list

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
75	objcl_ldr_user_data User program global data
76	objcl_ldr_sys_plabel System plabel table
77	objcl_user_nm_data User process NM global data
78	objcl_ldr_plabel_table Process plabel table for FINDPROC
79	objcl_gdpd_tbl Global data pointer descriptor table
80	objcl_fs_globals File system globals
81	objcl_fs_plfd_table Process local file descriptors
82	objcl_nm_error_stack NM diagnostic error stack
83	objcl_nm_heap NM heap
84	objcl_fio_state_sem_tbl File IO state semaphore table
85	objcl_pcb CM process control block table
86	objcl_pib Process information block table
87	objcl_pibx PIB extension table
88	objcl_boot_pib Boot process PIB
89	objcl_boot_pibx Boot process PIBX

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
90	objcl_processid_list Process ID list
91	objcl_protectid_list Protection ID list
92	objcl_system_map System map
93	objcl_pme_descriptor PME descriptor
94	objcl_pme_vat VAT from PME
95	objcl_pme_pit Page information table
96	objcl_pme_millicode Millicode from PME
97	objcl_pme_lst NL library symbol table
98	objcl_pme_sxrt System SXRT
99	objcl_progen_disc_page
100	objcl_mm_log
101	objcl_cme_desc
102	objcl_cme_abs
103	objcl_cme_cpool
104	objcl_cme_dpool
105	objcl_cme_pib
106	objcl_pme_real_code
107	objcl_pme_virtual_code
108	objcl_io_config Progen I/O configuration

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
109	objcl_progen_port Progen port
110	objcl_mm_safe_stack
111	objcl_io_buffer Load I/O buffers
112	objcl_disc_page_one Disk page one access
113	objcl_load_interlock Load interlock object
114	objcl_pdir Physical page directory
115	objcl_pdir_hdr PDIR table header
116	objcl_pdir_x Physical page directory extension
117	objcl_hash Physical page hash table
118	objcl_tcb Task control block
119	objcl_mm_ll Locality list table
120	objcl_mm_io_info
121	objcl_mm_io_notify
122	objcl_mm_io_req
123	objcl_mm_vs_ll Virtual space/LL headers table
124	objcl_mm_pp_hash
125	objcl_mm_ko_hash

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
126	objcl_mm_global_info
127	objcl_mm_grey_page
128	objcl_mm_tos_trap_port
129	objcl_mm_swapin_port
130	objcl_mm_io_port
131	objcl_mm_access Access object for initialization
132	objcl_mib_table Memory information block table
133	objcl_disp_global Dispatcher globals
134	objcl_disp_port Dispatcher port
135	objcl_disp_timer_port Dispatcher timer port
136	objcl_hpsql HP SQL process global objects
137	objcl_meas_kso_tbl Pointers for accessing counters
138	objcl_config_tbl Groups and counters per group
139	objcl_look_tbl Subclass to index table xref
140	objcl_index_tbl Table to group counters
141	objcl_counter_array Counter array table
142	objcl_unspecified_port Unspecified port

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
143	objcl_diagserv_temp Diag server temporary object
144	objcl_nm_store_restore
145	objcl_sr_candidate_list
146	objcl_sr_event_log
147	objcl_pfp_port Port facility process's port
148	objcl_srlat_vol_info
149	objcl_xm_page_cache_header XM page cache header object
150	objcl_xm_intrinsic_temp
151	objcl_cmports_iowait_compl_tbl CM ports I/O completion table
152	objcl_discutil_tmp_obj Temporary objects for discutil
153	objcl_amt AMT (Sampler) objects
154	objcl_sysmain_port
155	objcl_pfa_log
156	objcl_ptulog Procedure trace utility log
157	objcl_softdump Softdump objects
158	objcl_volcmds_tmp Volume commands temporary objects
159	objcl_volutil_tmp Volume utility temporary objects
160	objcl_tvi_mvt TVI mounted volume table

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
161	objcl_iotool I/O tools temporary objects
162	objcl_semlog_globals Semaphore tracing/logging globals
163	objcl_ext_int_table External interrupt table
164	objcl_timer_globals Timer globals
165	objcl_timer_table Timer table
166	objcl_diagmon_port Diagnostics monitor port
167	objcl_diagmon_buffer Diagnostics monitor buffer
168	objcl_diagsys_port Diagnostics process port
169	objcl_diagsys_buffer Diagnostics process buffer
170	objcl_diagmon_proc_tbl Diagnostic monitor process information table
171	objcl_hlio_sanctum HLIO sanctum
172	objcl_hlio_lpt Logical path table
173	objcl_hlio_rid_tbl I/O request table
174	Not used
175	objcl_hlio_tbuf_tbl Terminal buffer table

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
176	objcl_hlio_chain_tbl Data chain table
177	objcl_hlio_bucket_tbl Bucket table
178	objcl_hlio_cm_ioreq CM I/O request table
179	objcl_hlio_cm_state CM state table
180	objcl_hlio_zero Object of zeros to fill the disk
181-182	Not used
183	objcl_hlio_std_io_port Standard I/O port
184	objcl_hlio_ldm_port Kernel LDM port
185	objcl_hlio_mm_surrogate MM surrogate
186	Not used
187	objcl_hlio_mm_intrinsic MM intrinsic
188-190	Not used
191	objcl_hlio_mortician Mortician surrogate
192	objcl_hlio_op_surrogate Console messages surrogate
193	objcl_hlio_power_fail Powerfail surrogate
194	objcl_hlio_disc_pool LDM disk message pool
195	objcl_hlio_ldm_pool LDM non-disk message pool

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
196	objcl_syslog_globals System logging globals
197	objcl_cmports_object CM ports object
198	objcl_cmports_desc_obj CM ports descriptor object
199	objcl_cmports_dst CM ports port DST
200	objcl_cmports_dictionary CM ports dictionary
201	objcl_cmports_class_dict CM ports class name dictionary
202	objcl_cmports_hash_table CM ports hash table
203	objcl_cmports_nmcontext CM ports context area for NM compl
204	objcl_util_ics ICS
205	objcl_util_footprint Footprint table
206	objcl_llio_unknown Unknown LLIO object
207	objcl_llio_global_pool Global message pool
208	objcl_llio_ppt Physical path table
209	objcl_llio_iomont I/O monitor table
210	objcl_llio_global_ports Global port data areas

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
211	objcl_tmp_diag_logging Tmp diag logging table
212	objcl_cio_cam CIO CAM data areas
213	objcl_tmux_dam TMUX DAM data areas
214	objcl_terminal_dm Terminal DM data areas
215	objcl_hpib_dam HPIB DAM data areas
216	objcl_cs80_disc CS80 disk DM data areas
217	objcl_7978_tape HP 7978 tape DM data areas
218	objcl_pp_dm Page printer DM data areas
219	objcl_ciper_dm CIPER DM data areas
220	objcl_buffer_mgr Datacomm buffer manager
221	objcl_term_sp_dm AVESTA term DM
222	objcl_flow_control_mgr AVESTA flow control manager
223	objcl_damsel IEEE 8023 LAN card DAM
224	objcl_dume AVESTA download/upload manager
225	objcl_dcc_surrogate Datacomm configurator

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
226	objcl_nodal_mgr Datacomm nodal manager
227	objcl_dcl_vts_svr NS VT server
228	objcl_dcl_vts_ldm NS VT LDM
229	objcl_dcl_xp_sm4_data NS transport sm4 data area
230	objcl_dcl_xp_sm4_waitq NS transport sm4 wait queue
231	objcl_dcl_bmgr_port Buffer manager port
232	objcl_dcl_bmgr_global Buffer manager global tables
233	objcl_dcl_bmgr_lbpt Buffer manager logical pools
234	objcl_dcl_bmgr_sbpt Buffer manager share pools
235	objcl_dcl_bmgr_pbso Buffer manager pool segments
236	objcl_dcl_bmgr_pbo Buffer manager physical buffers
237	objcl_dcl_lnktrc_port Link trace manager port
238	objcl_dcl_lnktrc_global Link trace global tables
239	objcl_dcl_lnktrc_dynamic Link trace dynamic tables
240	objcl_srlat_global Store/Restore look-aside global

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
241	objcl_sr_globals Store/Restore process globals
242	objcl_sr_buffers Restore process buffers
243	objcl_break_uit Break user interrupt table
244	objcl_break_notify_q_port Break notify queue port
245	objcl_break_recv_int_port Break receive interrupt port
246	objcl_turbo_mmap_table Turbo NM map table
247	objcl_activ_indic Activity indicator port
248	objcl_cache_queue_cb_table VPN cache entry queue semaphore table
249	objcl_debug_aux Debug auxiliary object
250	objcl_debug_sys_sym_heap Debug system symbol heap
251	objcl_debug_proc_sym_heap Debug process symbol heap
252	objcl_debug_sys_lst_tree Debug sym lib symbol table tree
253	objcl_debug_hist Debug history stack
254	objcl_debug_cmd Debug command stack
255	objcl_debug_err Debug error stack

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
256	objcl_debug_mac Debug macro storage
257	objcl_debug_var Debug variable storage
258	objcl_dat_hist DAT history stack
259	objcl_dat_cmd DAT command stack
260	objcl_dat_err DAT error stack
261	objcl_dat_mac DAT macro storage
262	objcl_dat_var DAT variable storage
263	objcl_break_rbt Resident break table
264	objcl_dcl_bmgr_frames Buffer manager shared frame pool
265	objcl_dcl_bmgr_poolport Buffer manager pool port
266	objcl_compl_head_pool Interrupt completion head pool
267	objcl_operating_system_info Operating system information object
268	objcl_sysgen_misc_config_info Sysgen miscellaneous configuration information
269	objcl_working_set_log Working set measurement tool log
270	objcl_working_set_logging_cb Working set measurement tool CB

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
271	objcl_xm_globals Transaction management globals
272	objcl_pfp_port_control_block PFP port control block
273	objcl_std_process_ports_tbl Standard process ports table
274	objcl_unavailable Object class is unavailable
275	objcl_nm_subsys_lib NM subsystem library
276	objcl_vs_alloc_info VS allocation information CB
277	objcl_nm_error_stack_parms NM diagnostic error stack parms
278	objcl_rdb_tbl Remote debugger table
279	objcl_xm_dump_log Transaction management dump log
280	objcl_sna_imf_globals SNA/IMF globals
281	Not used
282	objcl_io_traps I/O traps pool
283	objcl_alink_dam Alink DAM data areas
284	objcl_eagl_disc_dm Eagle disk DM data areas
285	objcl_alink_host_dm Alink host DM data areas

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
286	objcl_wan_sdlc_driver SDLC wide area network driver
287	objcl_wan_latb_driver LATB wide area network driver
288	objcl_wan_rje_driver RLC wide area network driver
289	Not used
290	objcl_xm_cb_pool Transaction management cb pool
291	objcl_xm_sm_cb_array array of sm cbs for XM recovery
292	objcl_dcl_xp_ip_data Transport subsystem ip data
293	objcl_dcl_xp_ni_data Transport subsystem ni data
294	objcl_dcl_xp_map_tbl Transport subsystem map table
295	objcl_dcl_xp_tcp_dst Transport subsystem tcp DST
296	objcl_dcl_xp_trace Transport subsystem trace
297	objcl_bipc_timer_port File system IPC timer port
298	objcl_ssm_level_1_map SSM level 1 bit map
299	objcl_ssm_level_2_map SSM level 1 bit map
300	objcl_timer_sync_port Timer sync port

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
301	objcl_swtoem_name_cache Switch to CM name cache
302	objcl_bus_conv_mgr Bus converter manager data area
303	objcl_afi_dam Asynchronous ... interface
304	objcl_iotool_gm I/O tool general manager
305	objcl_iotool_wait_mgr I/O tool wait manager
306	objcl_workstation_cnfg_mgr Workstation configurator
307	objcl_dtsm Terminal
308	objcl_alcp AVESTA link control protocol
309	objcl_wan_diag_driver Wan diag driver
310	objcl_compl_entry_pool Completion entry pool
311	objcl_virtual_disc_mgr Virtual disk manager
312	objcl_psi_dump_surr PSI dump surrogate
313	objcl_vt Virtual terminal
314	objcl_rdb_dm RDB device manager
315	objcl_llio_known Known llio object

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
316	objcl_sp_spit Spooler process information table
317	objcl_sp_output_spfdir Output spoolfile directory
318	objcl_sp_input_spfdir Input spoolfile directory
319	objcl_sp_qheadtbl Table of output spoolfile queue head entries
320	objcl_sp_cmd_snapobj Snapshot information from either of the spfdirs, to be processed
321	objcl_mm_critical_pool Memory manager critical pool pages
322	objcl_cmports_port CM ports port
323	objcl_cmports_timer_tbl CM ports timer table
324	objcl_cmports_synonym CM ports synonym table
325	objcl_io_tbufs_pool I/O tbufs pool
281	Not used
327	objcl_htt Hardware traps table
328	objcl_htc Hardware traps common cells
329	objcl_hth Hardware traps header pool
330	objcl_htbm Debug bit maps pool

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
331	objcl_config_io Pool of config info cells
332	objcl_io_pquad_table I/O pquad pool
333	objcl_debug_table I/O debug table
334	objcl_scsi_dam SCSI DAM objects
335	objcl_scsi_ddm SCSI Disc DM objects
336	objcl_scsi_dds_dm SCSI DDS Dm objects
337	objcl_magneto_dm Magneto-optical (JAWS) DM objects
338	objcl_autochg_dm Autochanger (JAWS) DM objects
339	objcl_nio_alink_dam NIO alink DAM objects
340	objcl_lancelot_dam NIO LAN/Console DAM objects
341	objcl_nio_console_dm NIO Console DM objects
342-549	Not used

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
550	objcl_cm_user_code User code segments
551	objcl_cm_system_code System code segments
552	objcl_cm_user_data User data segments
553	objcl_system_data System data segments
554	objcl_cm_cstx CST extension block
555	objcl_cm_cst Code segment table
556	objcl_cm_dst Data segment table
557	objcl_cm_abs CM bank zero
558	objcl_jit Job information table
559	objcl_sys_jit System Job Information Table
560	objcl_jdt Job Directory Table
561	objcl_sys_jdt System Job Directory Table
562	objcl_serial_disc Serial Disk DST
563	objcl_buf_serial_disc Serial disk buffer
564	objcl_mail_message Mail intrinsic message DST

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
565	objcl_nls_table Native Language Support table
566	objcl_nls_language_dst NLS language DST
567	objcl_message_catalog_dir Message catalog directory
568	objcl_fs_shared_fcb_list File system shared FCB list
569	objcl_fs_control_block File System Control Block DST
570	objcl_user_logging User logging DST
571	objcl_lstt Logical Segment Transfer Table
572	objcl_port_dst PORT (IPC) DST
573	objcl_sl_reference_table SL Reference Table
574	objcl_cm_ldr_cache_segment Loader Cache Segment Table
575	objcl_cm_ldr_seg_tbl_ext Loader Segment Table Extention
576	objcl_cm_hpe_communication Initial/Progen communication
577	objcl_logical_dev_table Logical Device Table
578	objcl_device_class_table Device Class Table
579	objcl_tape_label_table Tape Label Table

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
580	objcl_lsttipc_table IPC Table
581	objcl_fmavt File Multi-Access Vector Table
582	objcl_association_table Device association table
583	objcl_loader_seg_table CM Loader Segment Table
584	objcl_system_plabel_table System Plabel table
585	objcl_dcl_ss_global NS session services globals
586	objcl_dcl_ss_dynamic NS session services dynamic tbls
587	objcl_dcl_sk_global NetIPC global tables
588	objcl_dcl_sk_process NetIPC process tables
589	objcl_dcl_xp_utiltemp NS transport temp DST
590	objcl_dcl_xp_paths NS transport PD, PDXs
591	objcl_dcl_xp_addr NS transport/NetIPC addr alloc
592	objcl_dcl_logtrc_dst log/trace global data segment
593	objcl_turbo_global_cb Turbo Image Globals
594	objcl_turbo_buffer_cb Turbo Buffers

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
595	objcl_turbo_user_cb Turbo User Control Block
596	objcl_turbo_remote_cb Turbo Remote Control Block
597	objcl_turbo_dscb_ext Turbo dataset CB Extension
598	objcl_turbo_sys_cb Turbo System Control Block
599	objcl_turbo_ilsr_cb Turbo ILR Control Block
600	objcl_turbo_maint_cb Turbo DBMAINT Control Block
601	objcl_turbo_abort_cb Turbo DBABORT Control Block
602-649	Not used
650	objcl_user_file_0
651	objcl_user_file_1
652	objcl_user_file_2
653	objcl_user_file_3
654	objcl_user_file_4
655	objcl_user_file_5

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
656	objcl_user_file_6
657	objcl_user_file_7
658	objcl_user_file_8
659	objcl_user_file_9
660	objcl_user_file_10
661	objcl_nm_sl
662	objcl_cm_sl
663	objcl_nm_program
664	objcl_cm_program
665	objcl_ord_fix
666	objcl_ord_var
667	objcl_ord_und
668	objcl_ord_byte_stream
669	objcl_ksam_fix
670	objcl_ksam_var
671	objcl_nm_ksam_fix
672	objcl_rio_fix
673	objcl_cir_fix
674	objcl_cir_var
675	objcl_cir_und
676	objcl_msg_fix
677	objcl_msg_var
678	objcl_msg_und
679	objcl_cm_fix
680	objcl_cm_var
681	objcl_cm_und
682	objcl_dir_root

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
683	objcl_dir_acct
684	objcl_dir_group
685	objcl_dir_hmdir
686	objcl_dir_user
687	objcl_dir_file
688	objcl_hpdb_1
689	objcl_hpdb_2
690	objcl_hpdb_3
691	objcl_hpdb_4
692	objcl_hpdb_5
693	objcl_hpdb_6
694	objcl_turbo_root TURBO Root file
695	objcl_turbo_data_set TURBO Data Set file
696	objcl_turbo_data_base_access TURBO Data Base Access file
697	objcl_turbo_dbabort_i_ TURBO DBABORT I-file
698	objcl_turbo_ilr_log TURBO ILR log file
699	objcl_turbo_dbrecov_restart TURBO DBRECOV Restart file
700	objcl_turbo_dbchange_file TURBO DBCHANGE file

Table C-3. Object Class Information for Large Map (continued)

Array Index	Object Class Name and Description
701-710	Not used
711	objcl_xm_log Transaction Management Log file
712	objcl_xm_residual_log_1 XM Residual log #1
713	objcl_xm_residual_log_2 XM Residual log #2
714	objcl_allbase_control_block_area Allbase control block area
715	objcl_spool_var
716	objcl_dep_log XLDCP log file, for Measurement Interface
717	objcl_xm_attach_to_log
718	objcl_xm_file_under_recovery
719	objcl_xm_actxid XM active trans obj for APR

Semaphore Class Information

This appendix provides information relating to semaphore class counters. All semaphore class counters are returned as an array of integers. The length of the array is 256. Each element in the array relates to a collection of semaphores collectively known as a semaphore class. The following table gives the semaphore class information corresponding to each index in the array. The array is assumed to start with an index of 0.

Table D-1. Semaphore Class Information

Array Index	Semaphore Class Name and Description
0	semcl_unknown_class
1	semcl_port_freeze_tbl Freeze entry table
2	semcl_non_res_desc_table Non-resident port desc table
3	semcl_res_port_desc_table Resident port descriptor table
4	semcl_wait_queue_table Global pool of waitq entries
5	semcl_local_desc_table Process local port desc table
6	semcl_port_control_block Port control block
7	semcl_port_message_pool Message pool for a port
8	semcl_std_process_ports_tbl Tbl for std process's ports
9	semcl_vs_od Virtual Space Object Desc
10	semcl_vs_shr_page_desc Shared Page Descriptor
11	semcl_vs_domain_desc Virtual Space Domain Desc
12	semcl_vs_global_b_tree Virtual Address Global B-Trees
13	semcl_vs_allocation Virtual Space Allocation
14	semcl_nm_debug_table XDB table

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
15	semcl_mpe_psd MPE pseudo-disable sync
16	semcl_mpe_cst MPE Code Segment Table
17	semcl_mpe_dst MPE Data Segment Table
18	semcl_dmcs Compare and Swap global sem
19	semcl_flock Semaphore for file locking
20	semcl_file_open Semaphore for FOPEN
21	semcl_gufd Global Unique File Descriptor
22	semcl_fs_gdpd Data Pointer Table Entry
23	semcl_fs_hash File System Hash Table
24	semcl_lab_tbl_cb Label Table Control Block Table
25	semcl_mvt_cb Mounted Volume Table Control Block
26	semcl_vsvidt_cb
27	semcl_cm_ports CM Ports tables
28	semcl_pm_cb Process Management Control Block
29	semcl_xm_lock XM synchronization semaphore

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
30	semcl_xm_lsrow_cplist_1 XM checkpoint list semaphore for logset log 1st half
31	semcl_xm_trls XM logset control blocks
32	semcl_xm_counter XM counter area
33	semcl_xm_error XM error area within sys log
34	semcl_xm_lsid XM logset dir in sys log set
35	semcl_xm_ufid XM UFID area within log set
36	semcl_xm_storeid XM store set area of sys log
37	semcl_ssm_dev_map
38	semcl_mi_lookup MI table semaphore
39	semcl_ldr_lft Loader file table
40	semcl_ldr_sys_plbltbl Ldr hash tbl for sys plabels
41	semcl_err_mgt Lock parms for error stack
42	semcl_vsm_miss_time VPN cache miss timer
43	semcl_vsm_tbl_vs_b_tree VS b-tree table
44	semcl_vsm_tbl_cache VPN cache table

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
45	semcl_vsm_tbl_ar_cache VPN cache table
46	semcl_vsm_tbl_vs_od VS object descriptor table
47	semcl_vsm_tbl_vs_od_gu_fd VS object desc/GU_FD table
48	semcl_vsm_tbl_dis_exp_vs_od VS disabled expand obj desc
49	semcl_vsm_tbl_ext_b_tree extent b-tree table
50	semcl_vsm_tbl_ext_ar_b_tree access rights b-tree table
51	semcl_vsm_tbl_sp_desc shared page descriptor table
52	semcl_vsm_tbl_domain_desc VS domain descriptor table
53	semcl_vsm_tbl_recovery VS recovery stack table
54	semcl_vsm_tbl_cache_queue_cb VPN cache queue cntrl blk tbl
55	semcl_turbo_global_cb
56	semcl_turbo_db_lock
57	semcl_turbo_dset_lock
58	semcl_turbo_pred_lock
59	semcl_turbo_buffer_cb
60	semcl_turbo_block_lock
61	semcl_turbo_relaccess
62	semcl_turbo_system_cb

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
63	semcl_xm_lsrow_dp XM protect XM log position ptr
64	semcl_xm_lsrow_io XM I/O pointer; sync flush of xm log for a logset
65	semcl_xm_lsrow_checkpoint
66	semcl_turbo_abort_wait
67	semcl_turbo_dbb_busywait
68	semcl_turbo_dbg_busywait
69	semcl_unknown_resource_class Unknown class passed into Obtain
70	semcl_pib_table
71	semcl_pcb_table
72	semcl_lru to prevent concurrent updating of the list of files (gufd) in lru order
73	semcl_ssm_global_area
74	semcl_turbo_reuser
75	semcl_turbo_bufaccess
76	semcl_turbo_opendset
77	semcl_turbo_putdel
78	semcl_turbo_user_busywait
79	semcl_pdir_x_global
80	semcl_sid_array_sema
81	semcl_indirect_locality_table
82	semcl_vpn_cache

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
83	semcl_obrel obtain/release semaphore
84	semcl_break_uit_tbl
85	semcl_rbt_header
86	semcl_rbt_entry
87	semcl_timer_table
88	semcl_cmports_desc_hdr
89	semcl_cmports_desc_entry
90	semcl_cmports_desc_tbl_hdr
91	semcl_cmports_dst
92	semcl_cmports_context_tbl_hdr
93	semcl_cmports_hash_entry
94	semcl_cmports_synonym_tbl_hdr
95	semcl_cmports_timer_tbl_hdr
96	semcl_cmports_iowait_compl_tbl_hdr
97	semcl_plfd_tbl_hdr
98	semcl_file_open_cnt_tbl
99	semcl_gdpd
100	semcl_gdpd_tbl_hdr
101	semcl_pdir_tbl_hdr
102	semcl_hlio_lpt
103	semcl_hlio_ioreq
104	semcl_hlio_cm_state protect CM Ldev State Array tbl
105	semcl_hlio_rid_tbl_hdr
106	semcl_hlio_ldm_port_tbl_hdr
107	semcl_hlio_cm_ioreq_tbl_hdr
108	semcl_hlio_fill_objs
109	semcl_hlio_objcl

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
110	semcl_js_debug
111	semcl_js_sys_debug
112	semcl_load_interlock to prevent concurrent updating of the information on the disc labels sector 0 & 1
113	semcl_ldr_plabel_tbl_hdr to prevent concurrent updating of the program label (plabel) table
114	semcl_ldr_pfl_tbl_hdr to prevent concurrent updating of the table containing the list of files used by process
115	semcl_ldr_unsat_tbl_hdr
116	semcl_mi_counter_array_tbl_hdr
117	semcl_mm_grey_page_tbl_hdr
118	semcl_mm_unused_tbl_hdr
119	semcl_mm_vs_ll_tbl_hdr
120	semcl_mm_io_req_tbl_hdr
121	semcl_mm_io_notify_tbl_hdr
122	semcl_mm_ll_tbl_hdr
123	semcl_mfw_cb
124	semcl_global_port_obj
125	semcl_timer_sync_port_pcb
126	semcl_syslog_globals
127	semcl_srlat_globals
128	semcl_stngt_lru
129	semcl_swtoem_name_cache_tbl_hdr
130	semcl_reply_info_tbl_hdr
131	semcl_sysmain_term_buf_tbl_hdr
132	semcl_mib_tbl_hdr
133	semcl_mvst_foct
134	semcl_mvst_open_count

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
135	semcl_mvst_vol_set
136	semcl_mvt_tbl_hdr
137	semcl_vsit_tbl_hdr
138	semcl_vsm_cache_queue_cb
139	semcl_wslog_cb
140	semcl_nm_debug_tbl_hdr
141	semcl_debug_aux_tbl_hdr
142	semcl_xm_pirow_ufile_list
143	semcl_xm_syslog_lsid
144	semcl_xm_syslog_ufile
145	semcl_xm_syslog_storeid
146	semcl_xm_record XM control block pool
147	semcl_xm_block
148	semcl_xm_process XM process block for XM lock list management.
149	semcl_xm_deadlock
150	semcl_xm_int_lsid semaphore synchronizes access access to the pool of logset entries in XM_STATIC_AREA.LOG SETID_POOL
151	semcl_xm_volume
152	semcl_xm_tape
153	semcl_xm_db_table
154	semcl_xm_lsrow_cplist_2 XM checkpoint list semaphore for logset log 2nd half
155	semcl_xm_trans_table XM sync XM_TRANSROW entries of XM globals area.
156	semcl_xm_syslog_error
157	semcl_errmgt

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
158	semcl_hlio_bucket_tbl_hdr
159	semcl_hlio_tbuf_tbl_hdr
160	semcl_io_config_tbl_hdr
161	semcl_symtbl_tbl_hdr
162	semcl_unknown_portcl_pcb
163	semcl_deleyed_reboot_pcb
164	semcl_cm_port_pcb
165	semcl_cmports_iowait_pcb
166	semcl_cmports_nm_iowait_pcb
167	semcl_cmports_dst_pool_pcb
168	semcl_llio_process_pcb
169	semcl_js_tmp_pcb
170	semcl_mfw_sig_pcb
171	semcl_restore_pcb
172	semcl_store_pcb
173	semcl_sr_surrogate_pcb
174	semcl_bnd
175	semcl_xm_log_post_initiator_pcb
176	semcl_xm_log_post_completor_pcb
177	semcl_xm_checkpoint_server_pcb
178	semcl_std_message_pcb
179	semcl_std_proc_int_pcb
180	semcl_std_signal_pcb
181	semcl_tvi_pcb
182	semcl_sysmain_reply_pcb
183	semcl_activ_indic_pcb
184	semcl_jobmain_port_pcb
185	semcl_sessionmain_port_pcb
186	semcl_jobqueue_port_pcb

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
187	semcl_jsmain_port_pcb
188	semcl_bipc_timer_port_pcb
189	semcl_progen_port_pcb
190	semcl_mm_tos_trap_port_pcb
191	semcl_mm_swapin_port_pcb
192	semcl_mm_io_port_pcb
193	semcl_disp_port_pcb
194	semcl_disp_timer_port_pcb
195	semcl_pfp_port_control_block_pcb
196	semcl_sysmain_port_pcb
197	semcl_hlio_ldm_port_pcb
198	semcl_hlio_disc_pool_pcb
199	semcl_hlio_ldm_pool_pcb
200	semcl_llio_global_pool_pcb
201	semcl_llio_global_ports_pcb
202	semcl_compl_head_pool_pcb
203	semcl_break_notify_q_port_pcb
204	semcl_hlio_mm_surrogate_pcb
205	semcl_hlio_mm_intrinsic_pcb
206	semcl_disp_pib Disp portion of PIB
207	semcl_disp_ready_q Ready list of processes
208	semcl_disp_globals
209	semcl_vsm_control_obj
210	semcl_vsm_disabled_expandable
211	semcl_semlog_spin_lock
212	semcl_xm_xidcounter
213	semcl_hlio_power_fail_pcb

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
214	semcl_llio_unknown_pcb
215	semcl_cio_cam_pcb
216	semcl_tmux_dam_pcb
217	semcl_terminal_dm_pcb
218	semcl_hpib_dam_pcb
219	semcl_cs80_disc_pcb
220	semcl_7978_tape_pcb
221	semcl_pp_dm_pcb
222	semcl_ciper_dm_pcb
223	semcl_alink_dam_pcb
224	semcl_eagl_disc_dm_pcb
225	semcl_buffer_mgr_pcb
226	semcl_wan_sdle_driver_pcb
227	semcl_wan_latb_driver_pcb
228	semcl_wan_rje_driver_pcb
229	semcl_term_sp_dm_pcb
230	semcl_flow_control_mgr_pcb
231	semcl_dune_pcb
232	semcl_dcc_surrogate_pcb
233	semcl_nodal_mgr_pcb
234	semcl_damsel_pcb
235	semcl_alink_host_dm_pcb
236	semcl_mm_sema mem mgr psdb semaphore
237	semcl_process_pri_sema Lock when boost/dec pri
238	semcl_timer_semaphore
239	semcl_xm_obj_hash_table
240	semcl_xm_obj_list_hash_table

Table D-1. Semaphore Class Information (continued)

Array Index	Semaphore Class Name and Description
241	semcl_pdir_spin_lock
242	semcl_prw_spin_lock
243	semcl_xm_spin_lock
244	semcl_ioppt_spin_lock
245	semcl_iomont_spin_lock
246	semcl_ioobj_spin_lock
247	semcl_iotraps_spin_lock
248	semcl_iopda_spin_lock
249	semcl_footprint_spin_lock
250	semcl_vpn_cache_spin_lock
251	semcl_demolish_process Locked when demolish process ports and cause_process_int
252	semcl_user_log_user_resources User Logging user area of LOGBUFF.
253	semcl_user_log_disc_resources User Logging LOGBUFF (disc related portions in commarea).
254	semcl_user_log_buffer_resources User Logging buffer area of LOGBUFF commarea and the buffer itself.
255	semcl_other

Global Counters

This appendix provides descriptions of the item numbers for the counters that measure system wide events as well as descriptions of the events measured. These item numbers are valid for **MIGLOBON** and **MIGLOBGET** calls.

Note

Counters associated with TurboIMAGE can be enabled, but they are not incremented.

Table E-1. Global Counter Information

Num	Counter Name (Data Type) and Description
21000	gmc_response_to_pt5 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of response times <.5 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).
21001	gmc_response_to_1 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of response times >=.5 secs and < 1 sec. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).
21002	gmc_response_to_1pt5 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of response times >= 1 sec and < 1.5 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).
21003	gmc_response_to_2 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of response times >= 1.5 secs and < 2 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).
21004	gmc_response_to_3 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of response times >= 2 secs and < 3 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21005	<p>gmc_response_to_4 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of response times ≥ 3 secs and < 4 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).</p>
21006	<p>gmc_response_to_5 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of response times ≥ 4 secs and < 5 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).</p>
21007	<p>gmc_response_to_10 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of response times ≥ 5 secs and < 10 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).</p>
21008	<p>gmc_response_to_20 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of response times ≥ 10 secs and < 20 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).</p>
21009	<p>gmc_response_over_20 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of response times ≥ 20 secs. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data from the user (another cr or enter).</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21010	<p>gmc_1st_response_to_pt5 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response < .5 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21011	<p>gmc_1st_response_to_1 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response >=.5 secs and < 1 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21012	<p>gmc_1st_response_to_1pt5 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response >= 1 sec and < 1.5 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21013	<p>gmc_1st_response_to_2 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response >= 1.5 sec and < 2 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21014	<p>gmc_1st_response_to_3 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response >= 2 sec and < 3 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21015	<p>gmc_1st_response_to_4 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response ≥ 3 sec and < 4 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21016	<p>gmc_1st_response_to_5 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response ≥ 4 secs and < 5 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21017	<p>gmc_1st_response_to_10 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response ≥ 5 secs and < 10 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21018	<p>gmc_1st_response_to_20 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response ≥ 10 secs and < 20 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>
21019	<p>gmc_1st_response_over_20 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of first response ≥ 20 secs. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. This time is measured within the system before the write is sent to the terminal.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21020	gmc_think_to_1 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of think times < 1 sec. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.
21021	gmc_think_to_2 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of think times ≥ 1 sec and < 2 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.
21022	gmc_think_to_3 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of think times ≥ 2 secs and < 3 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.
21023	gmc_think_to_4 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of think times ≥ 3 secs and < 4 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal (waiting for data (cr, enter)), until HLIO receives a reply.
21024	gmc_think_to_5 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of think times ≥ 4 secs and < 5 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.
21025	gmc_think_to_10 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of think times ≥ 5 secs and < 10 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21026	<p>gmc_think_to_20 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of think times ≥ 10 secs and < 20 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.</p>
21027	<p>gmc_think_to_30 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of think times ≥ 20 secs and < 30 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.</p>
21028	<p>gmc_think_to_40 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of think times ≥ 30 secs and < 40 secs. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply.</p>
21029	<p>gmc_think_over_40 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is valid only for terminal device types. Think time is defined by the system as the time beginning when a HLIO sends a read request to the terminal waiting for data (cr, enter), until HLIO receives a reply. The total number of think times ≥ 40 secs.</p>
21030	<p>gmc_min_think_time (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This is the minimum think time threshold value. This threshold is provided to allow users to filter transactions. The goal is to eliminate status reads from being accumulated without excluding other real transactions. This value is currently set at 100 ms.</p>
21031	<p>gmc_max_think_time (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This is the maximum think time threshold value. This threshold is provided to allow users to filter transactions. The goal is to prevent extremely long transactions from being accumulated. This value is currently set at MAXINT.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21032	<p>gmc_think_under_min_threshold (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>Number of read completions with think time under the minimum threshold. When the think time is found to be under the threshold, the time for this transaction continues to be accumulated as if nothing had occurred. This ensures that if a status read has occurred, the time spent processing that read will be accumulated as response time. If the think time is greater than the minimum threshold and under the maximum threshold, the counters for the transaction are measured.</p>
21033	<p>gmc_think_over_max_threshold (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>Number of read completions with think time over the maximum threshold. When the think time is found to be over the threshold, the transaction is not measured. If the think time is greater than the minimum threshold and under the maximum threshold, the counters for the transaction are measured.</p>
21034	<p>gmc_tot_buffer_access (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of TurboIMAGE buffer accesses.</p>
21035	<p>gmc_buffer_overlay (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a TurboIMAGE buffer is overlaid.</p>
21036	<p>gmc_buffer_hits (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the required buffer is in the TurboIMAGE buffer pool.</p>
21037	<p>gmc_buffer_full (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the required buffer is not found in the TurboIMAGE buffer pool.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21038	gmc_buffer_cnt_change (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21039	gmc_buffer_pri_readj (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21040	gmc_find_on_putdelete_q (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times DBFIND waits on DBPUT/DBDELETE queue.
21041	gmc_blocks_ser_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for serial read.
21042	gmc_blocks_cal_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for calculated read.
21043	gmc_blocks_put_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for DBPUT master.
21044	gmc_write_put_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_write calls for DBPUT detail.
21045	gmc_write_del_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_write calls for DBDELETE detail.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21046	gmc_tot_put_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBPUT detail.
21047	gmc_tot_del_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBDELETE detail.
21048	gmc_tot_ser_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of Serial read calls.
21049	gmc_tot_cal_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of calculated read calls.
21050	gmc_tot_put_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBPUT master.
21051	gmc_prim_hash_collision (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of primary hash collisions.
21052	gmc_dirt_buffer_overlay (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a dirty TurboIMAGE buffer is overlaid by the process.
21053	gmc_pred_lock_collision (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of predicate lock collisions (for >= <= > and <).

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21054	gmc_blocks_chain_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for chain read.
21055	gmc_write_put_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_write calls for DBPUT master.
21056	gmc_write_del_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_write calls for DBDELETE master.
21057	gmc_wlabel_put_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21058	gmc_wlabel_del_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21059	gmc_tot_del_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBDELETE master.
21060	gmc_tot_det_paths (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21061	gmc_tot_dbbegin (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBBEGIN calls.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21062	gmc_tot_dbcontrol (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBCONTROL calls.
21063	gmc_tot_dbend (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBEND calls.
21064	gmc_tot_dbfind (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBFIND calls.
21065	gmc_tot_dbget (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBGET calls.
21066	gmc_tot_DBinfo (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBINFO calls.
21067	gmc_tot_dblock (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBLOCK calls.
21068	gmc_tot_dbopen (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBOPEN calls.
21069	gmc_tot_dbunlock (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBUNLOCK calls.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21070	gmc_tot_dbupdate (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBUPDATE calls.
21071	gmc_tot_chain_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for chain read.
21072	gmc_tot_dbput (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBUPDATE calls.
21073	gmc_tot_dbdelete (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBDELETE calls.
21074	gmc_pri_demote_3to2 (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21075	gmc_pri_demote_2to1 (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21076	gmc_pri_demote_1to0 (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21077	gmc_tot_dbclose (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBCLOSE calls.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21078	gmc_tot_dberror (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBERROR calls.
21079	gmc_tot_dbexplain (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBEXPLAIN calls.
21080	gmc_tot_dbmemo (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBMEMO calls.
21081	gmc_ipc_transfer_to_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a process is sent from a process environment to a port whose server can only be invoked on the ICS.
21082	gmc_pfp_port_locked_on_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the port facility process was invoked due to an attempt to access a port on the ICS when it was already locked.
21083	gmc_pfp_pool_locked_on_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of times the port facility process was invoked due to an attempt to access a port's message pool on the ICS when it was already locked.
21084	gmc_pfp_cannot_access_port_from_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of times that the port facility process was invoked due to an attempt to invoke a port's procedure server that is not allowed to execute on the ICS.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21085	<p>gmc_pfp_msg_threshold_exceeded (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the port facility process was invoked when the number of messages queued on a port exceeded the maximum threshold. This is to avoid a process that does a send to be stuck as the target stack for multiple invocations of the procedure server.</p>
21086	<p>gmc_ics_partial_make_present (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times there was a send access to a port that is non-resident but that is ics-sendable, and therefore the port/pool must be made present in memory before the send access may be completed.</p>
21087	<p>gmc_ics_total_make_present (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times there was a send access to a port that is non-resident but that is ics-sendable and whose procedure server is ics-executable; therefore the port, pool, server data area, code, and any auxiliary data areas must be made present in memory before the server may be invoked.</p>
21088	<p>gmc_semcl_que_1_3 (I32A) Counter Type: Semaphore Class Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks on a semaphore and finds the queue length of that semaphore ≥ 1 and ≤ 3. Note that the counter measures this event for a semaphore class that consists of a collection of semaphores.</p>
21089	<p>gmc_semcl_que_4_6 (I32A) Counter Type: Semaphore Class Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks on a semaphore and finds the queue length of that semaphore ≥ 4 and ≤ 6. Note that the counter measures this event for a semaphore class that consists of a collection of semaphores.</p>
21090	<p>gmc_semcl_que_7_over (I32A) Counter Type: Semaphore Class Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks on a semaphore and finds the queue length of that semaphore ≥ 7. Note that the counter measures this event for a semaphore class that consists of a collection of semaphores.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21091	gmc_cum_ready_queue (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes entered the ready queue to be serviced by the CPU since measurements were enabled.
21092	gmc_cur_ready_queue (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of processes currently in the ready queue waiting to be serviced by the CPU (the length of the ready queue).
21093	gmc_max_ready_queue (I32) Counter Type: Simple Counter Unit of Measurement: Count The maximum length (high-water mark) of the ready queue since measurements were enabled.
21094	gmt_cum_request_secs (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total time spent in the ready queue by every process that entered the ready queue (also known as the system residence time). If a graph is drawn between the number of processes in the ready queue and time, this gives the area of the graph.
21095	gmt_ready_q_0 (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This gives the length of time during which there was no process in the ready queue. Could be interpreted as idle time in the CPU.
21096	gmt_ready_q_1 (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This gives the length of time during which there was only one process in the ready queue.
21097	gmt_ready_q_2 (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This gives the length of time during which there were two processes in the ready queue.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21098	gmt_ready_q_4 (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This gives the span of time during which the length of the ready queue was between 3 and 4.
21099	gmt_ready_q_8 (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This gives the span of time during which the length of the ready queue was between 4 and 8.
21100	gmt_ready_q_16 (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This gives the span of time during which the length of the ready queue was between 8 and 16.
21101	gmt_ready_q_32 (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This gives the span of time during which the length of the ready queue was between 16 and 32.
21102	gmt_ready_q_over_32 (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This gives the length of time during which there were more than thirty-two processes in the ready queue.
21103	gmc_cur_blk_nm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a native mode code page fault.
21104	gmc_cur_blk_nm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a native mode stack page fault.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21105	gmc_cur_blk_nm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a native mode transient page fault (heap, swappable table).
21106	gmc_cur_blk_file_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a file page fault.
21107	gmc_cur_blk_cm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a compatibility mode code page fault.
21108	gmc_cur_blk_cm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a compatibility mode stack page fault.
21109	gmc_cur_blk_cm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for an I/O to complete on a compatibility mode transient page fault.
21110	gmc_cur_blk_trm_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a terminal read.
21111	gmc_cur_blk_trm_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a terminal write such as console I/O or general message for WARN and TELL messages.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21112	gmc_cur_blk_disc_io (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for I/O on discs.
21113	gmc_cur_blk_io_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for I/O on non-disc devices.
21114	gmc_cur_blk_ipc_trans_complete (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that blocked on IPC with transaction completed as one of their options(for example, the IOWAIT intrinsic).
21115	gmc_cur_blk_sir_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a SIR.
21116	gmc_cur_blk_rin_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a RIN.
21117	gmc_cur_blk_mem_mgr_prefetch (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a memory manager prefetch.
21118	gmc_cur_blk_quantum_expiration (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21119	gmc_cur_blk_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a timeout or pause with one second or less.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21120	gmc_cur_blk_parent_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are waiting for their parents to wake them up.
21121	gmc_cur_blk_cntl_block_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a control block on a semaphore.
21122	gmc_cur_blk_child_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for their children to wake them up.
21123	gmc_cur_blk_data_comm_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for data communication.
21124	gmc_cur_blk_rit_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are waiting for a RIT wait (operator reply)
21125	gmc_cur_blk_disp_work (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21126	gmc_cur_blk_port_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a port (default IPC wait).

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21127	gmc_cur_blk_mail_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a port for a MAIL (old type of IPC interface that existed prior to message file implementation).
21128	gmc_cur_blk_junk_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a port for a JUNK wait (don't care type wait).
21129	gmc_cur_blk_message_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a port for a MESSAGE (basic IPC message file).
21130	gmc_cur_blk_impede (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are impeded (mostly used by the file system for synchronization purposes).
21131	gmc_cur_blk_break_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked in break mode.
21132	gmc_cur_blk_wait_queue (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are being put on a wait queue by PORTS (table management running out of entries waits for additional spaces to be allocated).
21133	gmc_cur_blk_memmgt_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of times that the memory manager blocked for proper I/O synchronization, excluding user I/O request such as POST.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21134	gmc_cur_blk_port_blocked_make_present (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of process that are blocked because PORTS needs to wait until the requested port is present.
21135	gmc_cur_blk_file_blocked (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a port when posting pages through the call CM_POST.
21136	gmc_cur_blk_file_unblocked (I32) Counter Type: Simple Counter Unit of Measurement: Count This block on port not used. This counter should return zero.
21137	gmc_cur_blk_storage_mgt (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of process that are blocked on a port through storage management.
21138	gmc_cur_blk_user_to_debug_msg (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes(CIs) that are blocked due to breakpoint contention.
21139	gmc_cur_blk_io_config_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked because devices are being configured or released.
21140	gmc_cur_blk_pfp_reply_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked because the port facility process needs to be created, initialized, or checked.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21141	gmc_cur_blk_db_monitor_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked because the database monitor (SQL) is waiting for the DB CLEAN_UP process to finish cleaning up the aborted processes before closing the database.
21142	gmc_cur_blk_fill_disc_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked because HLIO is waiting for the master MIB because the new file extent in secondary storage needs to be initialized with fill characters for all virgin pages.
21143	gmc_cur_blk_hlio_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked because the HLIO is aborting the I/O.
21144	gmc_cur_blk_fs_tio_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked because Terminal I/O (TIO) fast write in DTS (BND LDM) is waiting for a reply message from the device manager or buffer management.
21145	gmc_cur_blk_mem_mgr_post_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked waiting for I/O completion when explicitly posting pages to disk.
21146	gmc_cur_blk_signal_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked for a delay or timer on a standard signal port.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21147	gmc_cur_blk_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21148	gmc_cur_blk_disc_io_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21149	gmc_cur_blk_priority_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21150	gmc_cur_blk_sql_lock_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked to acquire a SQL lock. This lock is required for user data (a tuple, a page, or a relation).
21151	gmc_cur_blk_sql_latch_level_1_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a level 1 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).
21152	gmc_cur_blk_sql_latch_level_2_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a level 2 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).
21153	gmc_cur_blk_sql_latch_level_3_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a level 3 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21154	gmc_cur_blk_sql_latch_level_4_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked on a level 4 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).
21155	gmc_cur_blk_sql_buffer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are blocked to change the state of a page in the SQL buffer. The buffer is used to hold a page (4 K) of user data. The page can be in a number of states (for example, being updated, in transit in, in transit out of) When a process requests that a page be placed in a state that conflicts with its current state, the process blocks.
21156	gmc_cur_blk_long_pause_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes that are currently paused for 2 secs or more.
21157	gmc_cur_blk_mem_mgr_freeze_and_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes currently blocked on freeze and corner cases other than page_fault, prefetch, and freeze. This counter would be predominantly block on freeze.
21158	gmc_cur_blk_wait_other (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
21159	gmc_cur_as_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes in the AS queue.
21160	gmc_cur_bs_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes in th BS queue.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21161	gmc_cur_cs_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes in the CS queue.
21162	gmc_cur_ds_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes in the DS queue.
21163	gmc_cur_es_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of processes in the ES queue.
21164	gmc_pauses (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the dispatcher enters the paused state. This state is reached when there are no ready processes and there is at least one process blocked on disk io.
21176	gmt_paused (I32) Counter Type: Simple Counter Unit of Measurement: ML Time The total length of time that the dispatcher was in the paused state. This state is reached when there are no ready processes and there is at least one process blocked on disk I/O.
21246	gmc_nm_pgms_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This represents the total number of program files that the loader has loaded for all processes since measurements were started. This counter is never decremented.
21247	gmc_nm_libs_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This represents the total number of library files that the loader has loaded for all processes since measurements were started. This counter is never decremented.
21248	gmc_cm_pgms_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
21249	gmc_cm_libs_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21250	gmc_cur_nm_pgms_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This represents the current number of program files that are loaded for all processes on the system.
21251	gmc_cur_nm_libs_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This represents the current number of library files that are loaded for all processes on the system.
21252	gmc_cur_cm_pgms_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
21253	gmc_cur_cm_libs_alloc (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
21254	gmc_page_ft_count_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the total number of page faults.
21255	gmc_page_ft_len_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the total number of pages brought in on a page fault due to an implicit prefetch.
21256	gmc_page_ft_on_virgin_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the total number of virgin page faults. A virgin page is a page that has never before been referenced. Therefore, it should not be read from disk, but should be filled with the initialization character instead. This is an optimization to prevent unnecessary disk reads.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21257	gmc_memmgt_unblocked_virgin_page_ft_group (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the number of virgin page faults that could be satisfied from the main memory free list without replacing any pages.
21258	gmc_memmgt_unblocked_virgin_page_ft_pages_group (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that were allocated from the main memory free list without replacing any pages for a virgin page fault.
21259	gmc_page_ft_io_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that were brought to main memory due to I/O page faults that resulted in I/Os.
21260	gmc_page_ft_io_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBS (actual disk I/O requests) generated due to I/O page faults.
21261	gmc_prefetch_count_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the total number of explicit prefetches (reads) from other than the memory manager.
21262	gmc_prefetch_len_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the total number of pages in each prefetch request.
21263	gmc_prefetch_all_virgin_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count This counter counts the number of performed prefetches of which all pages requested were virgin.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21264	gmc_prefetch_all_virgin_wait_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of prefetches of which all pages requested were virgin but could not be allocated from the current main memory free list. The memory manager has to replace a few pages in main memory to satisfy this request.
21265	gmc_prefetch_all_virgin_nowait_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of prefetches of which all pages requested were virgin and could be allocated from the current main memory free list without replacement.
21266	gmc_prefetch_unblocked_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of prefetches that were done for which the process did not block until all the pages were brought in.
21267	gmc_prefetch_io_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages in prefetch for which actual I/O was done.
21268	gmc_prefetch_io_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBS (actual I/O requests) generated by prefetch.
21269	gmc_post_count_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of times explicit post (writes) was called by processes to post pages to disk.
21270	gmc_post_len_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages in post requests.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21271	gmc_post_unblocked_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of times that explicit post was called and the process did not block until all the pages were written to disk.
21272	gmc_post_io_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages in post for which actual I/O was done.
21273	gmc_post_io_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBS (actual I/O requests) generated by post.
21274	gmc_post_make_oc_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of times a post to disk is due to making some pages an overlay candidate. This happens when some pages are no longer needed in memory and should be made ROCs.
21275	gmc_post_make_oc_len_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that were made ROCs and posted to disk.
21276	gmc_make_oc_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages made ROCs explicitly, for example, the storage manager. etc.
21277	gmc_make_oc_lru_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages made ROCs implicitly, for example, by the LRU algorithm used by the memory manager.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21278	gmc_make_dirty_oc_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that were dirty and made ROCs.
21279	gmc_make_oc_io_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that were written to disk when a page was made ROC and found to be dirty. In such cases, all contiguous virtual pages before and after the virtual page that was made an ROC and dirty are written to disk.
21280	gmc_make_oc_io_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBs (actual I/O requests) generated by dirty pages made Roc. (includes gather writes). In such cases all contiguous virtual pages before and after the virtual page that was made an ROC and dirty are written to disk.
21281	gmc_block_kickout_oc_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of times that processes blocked until a kicked out page was written to disk.
21282	gmc_recover_oc_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that were made an overlay candidate (put in the ROC list) and were brought to memory when referenced.
21283	gmc_overlay_oc_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages in the ROC list that were given to other virtual pages (overlaid).
21284	gmc_frz_pg_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of times pages in an object are frozen.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21285	gmc_cur_frz_pg_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that are currently frozen in main memory.
21286	gmc_cur_res_pg_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The current number of pages that are currently resident in main memory.
21287	gmc_ref_ko_page_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that have been referenced before being kicked out.
21288	gmc_unknown_write_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that are written by explicit calls other than post.
21289	gmc_unknown_write_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBs (actual I/O requests) generated by explicit write calls other than post.
21290	gmc_unknown_read_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages that are read by explicit calls other than prefetch.
21291	gmc_unknown_read_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBs (actual I/O requests) generated by explicit read calls other than post.
21292	gmc_bkgrnd_wt_io_pages_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of pages written out due to background write. When the CPU is idle, it goes through the main memory and writes dirty pages to disk.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21293	gmc_bkgrnd_wt_io_mibs_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of MIBS (actual I/O requests) due to background writes. When the CPU is idle, it goes through the main memory and writes dirty pages to disk.
21294	gmc_cur_swap_pg_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The current number of pages in main memory that can be replaced (swappable, that is, non-frozen and non-resident).
21295	gmc_cum_alloc_main_mem (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of virtual pages that have been allocated main memory.
21296	gmc_mem_add_ko_page_to_loc (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages kicked out of memory that are added to a process locality list.
21297	gmc_mem_ko_insert (I32) Counter Type: Simple Counter Unit of Measurement: Count Th total number of times a virtual page is inserted into the kick-out hash table.
21298	gmc_mem_ko_collision (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a virtual page is found in the kick-out hash table.
21299	gmc_mem_pp_insert (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a virtual page is inserted into the pseudo-present (PP) hash table.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21300	gmc_mem_pp_collision (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a virtual page is found in the pseudo-present hash table.
21301	gmc_tos_trap_1 (I32) Counter Type: Simple Counter Unit of Measurement: Count
21302	gmc_tos_trap_2 (I32) Counter Type: Simple Counter Unit of Measurement: Count
21303	gmc_tos_trap_3 (I32) Counter Type: Simple Counter Unit of Measurement: Count Time for quick path.
21304	gmc_mem_add_to_loc_good_guess (I32) Counter Type: Simple Counter Unit of Measurement: Count
21305	gmc_mem_locality_list_full (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the memory manager finds the process locality list full.
21306	gmc_mem_clock_cycles (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of revolutions through memory (LRU algorithm).
21307	gmc_mem_pressure (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the memory manager detects memory pressure.
21308	gmc_cum_alloc_free_main_mem (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages put in the main memory free list.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21309	gmc_cur_free_main_mem_pg (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of pages currently in the main memory free list.
21310	gmc_cur_roc_pages (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of recoverable overlay candidates (number of pages in the ROC list).
21311	gmc_cum_roc_pages (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times virtual pages were put on the ROC list.
21312	gmc_mem_swappable_pages_examined (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages examined by the LRU algorithm.
21313	gmc_sw2nm_count (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the system went from compatibility mode (CM) to native mode (NM) and back.
21314	gmc_sw2cm_count (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the system went from native mode (NM) to compatibility mode (CM) and back.
21315	gmc_sw2cm_wrapdst (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that a CM data segment was mapped on top of an NM object so that the specified portion of the object can be accessed using data segment access methods. This is usually done for split stacks.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21318	gmc_cum_pgm_completions (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of program completions.
21319	gmc_cur_sys_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of system processes created.
21320	gmc_cur_ci_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of CI processes created.
21321	gmc_cur_ci_created_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of user processes created by CI.
21322	gmc_cur_non_ci_created_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of user processes created by another user's process.
21323	gmc_sm_dummy_counter (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is a dummy used to reserve MI space for testing new counters; it is not currently in use.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21324	gmc_sm_r_trans_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented.
21325	gmc_sm_w_trans_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented.
21326	gmc_sm_r_random_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever a page required by a random access read is not in memory and must be brought in to complete the read.
21327	gmc_sm_w_random_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever a page required by a random access write is not in memory and must be brought in to complete the write.
21328	gmc_sm_r_ss_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented.
21329	gmc_sm_w_ss_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented.
21330	gmc_sm_r_se_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever a page required by a sequential access read is not in memory and must be brought in to complete the read.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21331	gmc_sm_w_se_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever a page required by a sequential access write is not in memory and must be brought in to complete the write.
21332	gmc_sm_r_fetch_too_much (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented in disk storage management.
21333	gmc_sm_w_fetch_too_much (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented in disk storage management.
21334	gmc_sm_read_count (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of disk file reads performed on the system.
21335	gmc_sm_write_count (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of disk file writes performed on the system.
21336	gmc_sm_hash_hit (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever a new GUFID entry hashes to an empty file hash table entry; for optimal system performance, this number should greatly exceed the gmc_sm_hash_miss counter.
21337	gmc_sm_hash_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever a new GUFID entry collides in the file hash table with an existing GUFID entry; when this occurs, the new GUFID must be placed on the existing hash entry's linked list. For optimal system performance, this number should be much smaller than the gmc_sm_hash_hit counter.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21338	<p>gmc_sm_hash_depth (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the high-water mark of the file system hash table collision lists. The higher this number, the worse file access performance will be.</p>
21339	<p>gmc_sm_lru_depth (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of GUFD entries currently linked to the least recently used closed file list (LRU). The gmc_sm_lru_depth counter is incremented when a closed file's GUFD entry is added to the LRU; it is decremented when a GUFD entry is removed from the LRU (when the gmc_sm_gufd_reused counter is incremented).</p>
21340	<p>gmc_sm_gufd_reused (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is incremented whenever a GUFD entry is removed from the LRU list (least recently used closed file list), and reused for another file.</p>
21342	<p>gmc_sm_read_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file reads requested when the page to be read is not in memory and thus must be fetched from disk to complete the read.</p>
21344	<p>gmc_sm_write_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file writes requested when the page to be written is not in memory and must be fetched from disk to complete the write.</p>
21345	<p>gmc_sm_r_mp_too_much (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file read requests where the page to be read was earlier brought into memory in anticipation of a likely read request, but the page was swapped out due to memory pressure before the read request actually occurred.</p>
21346	<p>gmc_sm_w_mp_too_much (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file write requests where the page to be written was earlier brought into memory in anticipation of a likely write request, but the page was swapped out due to memory pressure before the write request actually occurred.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21347	gmc_sm_r_fetch_again (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented in disk storage management.
21348	gmc_sm_w_fetch_again (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented in disk storage management.
21349	gmc_sm_r_begin_soft_pf (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented in disk storage management; it was replaced with gmc_sm_r_soft_pf.
21350	gmc_sm_w_begin_soft_pf (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not currently implemented in disk storage management; it was replaced with gmc_sm_w_soft_pf.
21351	gmc_sm_r_soft_pf (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter records the number of soft prefetches done on behalf of a disk file read request. A soft prefetch is a fetch of a disk file page into memory in anticipation of an upcoming read/write request, although that page has not yet been requested for any currently outstanding read/write.
21352	gmc_sm_w_soft_pf (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter records the number of soft prefetches done on behalf of a disk file write request. Note: a soft prefetch is a fetch of a disk file page into memory in anticipation of an upcoming read or write request, although that page has not yet been requested for any currently outstanding read or write.
21353	gmc_sm_w_vp_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter records the number of disk file writes that would have required a memory fetch from disk, but which utilize virgin prefetching.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21354	<p>gmc_sm_r_imi_too_late (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file reads where the page to be read is already in motion into memory, but the read must still wait for the page to be fully brought into memory.</p>
21355	<p>gmc_sm_w_imi_too_late (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file writes where the page to be written is already in motion into memory, but the write must still wait for the page to be fully brought into memory.</p>
21356	<p>gmc_sm_adj_fetch_amount (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is incremented whenever the number of pages to be prefetched and the page look ahead buffer is adjusted either up or down.</p>
21357	<p>gmc_sm_adj_fetch_ahead (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is not currently implement in disk storage management; it was replaced with the gmc_sm_adj_fetch_amount counter.</p>
21358	<p>gmc_sm_open_new_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is incremented whenever a new disk file is created.</p>
21359	<p>gmc_sm_open_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk files opened on the system. Please note: this is not the number of files currently open on the system; it is the number of files opened from the time measurement began. To calculate the number of file currently open, subtract the gmc_sm_close_file count.</p>
21360	<p>gmc_sm_close_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk file closes performed on the system since measurement began.</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21361	gmc_sm_purge_file (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented whenever any disk file is purged from the system.
21362	gmc_sm_random_read (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the total number of random access disk file reads requested on the system since measurement began.
21363	gmc_sm_random_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the total number of random access disk file writes requested on the system since measurement began.
21364	gmc_sm_seq_read (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the total number of sequential access disk file reads requested on the system since measurement began.
21365	gmc_sm_seq_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the total number of sequential access disk file writes requested on the system since measurement began.
21366	gmc_t_space_allocation_requests (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of transient pages on disk requested for space allocation.
21367	gmc_p_space_allocation_requests (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of permanent pages on disk requested for space allocation.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21368	gmc_t_pages_allocated (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of transient pages on disk allocated.
21369	gmc_p_pages_allocated (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of permanent pages on disk allocated.
21370	gmc_t_pages_not_got (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of transient pages that were preferred but did not get allocated (the difference between the number of pages preferred and the number of pages allocated).
21371	gmc_p_pages_not_got (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of permanent pages that was preferred but did not get allocated (the different between the number of pages preferred and the number of pages allocated).
21372	gmc_t_total_failures (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of failures while trying to satisfy the requests for transient space allocation.
21373	gmc_p_total_failures (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of failures while trying to satisfy the requests for permanent space allocation.
21374	gmc_t_dev_checked (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of devices checked while trying to satisfy the requests for transient space allocations.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21375	gmc_p_dev_checked (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of devices checked while trying to satisfy the requests for permanent space allocations.
21376	gmc_t_dev_searched (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of devices searched while trying to satisfy the requests for transient space allocations.
21377	gmc_p_dev_searched (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of devices searched while trying to satisfy the requests for permanent space allocations.
21378	gmc_t_alloc_dpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page faults while allocating transient space on disk.
21379	gmc_p_alloc_dpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page faults while allocating permanent space on disk.
21380	gmc_t_alloc_cpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults while allocating transient space on disk.
21381	gmc_p_alloc_cpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults while allocating permanent space on disk.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21382	gmc_t_deallocs (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that transient space on disk was deallocated.
21383	gmc_p_deallocs (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that permanent space on disk was deallocated.
21384	gmc_t_pages_dealloc (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages in transient space that were deallocated.
21385	gmc_p_pages_dealloc (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages in permanent space that were deallocated.
21386	gmc_t_dealloc_dpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page faults while deallocating transient space.
21387	gmc_p_dealloc_dpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page faults while deallocating permanent space.
21388	gmc_t_dealloc_cpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults while deallocating transient space.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21389	gmc_p_dealloc_cpage_fts (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults while deallocating permanent space.
21390	gmc_cur_sec_storage_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to transient data objects.
21391	gmc_cur_sec_storage_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to permanent data objects.
21392	gmc_cur_sec_storage_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to NM stack objects.
21393	gmc_cur_sec_storage_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to CM stack objects.
21394	gmc_cur_sec_storage_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to NM code objects.
21395	gmc_cur_sec_storage_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to CM code objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21396	gmc_cur_sec_storage_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to CM data objects.
21397	gmc_cur_sec_storage_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to file objects.
21398	gmc_cur_sec_storage_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to NM system library objects.
21399	gmc_cur_sec_storage_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of secondary storage on disk allocated to CM system library objects.
21400	gmc_cum_sec_storage_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to transient data objects.
21401	gmc_cum_sec_storage_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to permanent data objects.
21402	gmc_cum_sec_storage_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to NM stack objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21403	gmc_cum_sec_storage_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to CM stack objects.
21404	gmc_cum_sec_storage_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to NM code objects.
21405	gmc_cum_sec_storage_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to CM code objects.
21406	gmc_cum_sec_storage_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to CM data objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21407	gmc_cum_sec_storage_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to file objects.
21408	gmc_cum_sec_storage_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to NM system library objects.
21409	gmc_cum_sec_storage_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to CM system library objects.
21410	gmc_cur_virt_mem_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to transient data objects.
21411	gmc_cur_virt_mem_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to permanent data objects.
21412	gmc_cur_virt_mem_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to NM stack objects.
21413	gmc_cur_virt_mem_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to CM stack objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21414	gmc_cur_virt_mem_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to NM code objects.
21415	gmc_cur_virt_mem_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to CM code objects.
21416	gmc_cur_virt_mem_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to CM data objects.
21417	gmc_cur_virt_mem_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to file objects.
21418	gmc_cur_virt_mem_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to NM system library objects.
21419	gmc_cur_virt_mem_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of bytes of virtual memory allocated to CM system library objects.
21420	gmc_cum_virt_mem_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to transient data objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21421	gmc_cum_virt_mem_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to permanent data objects.
21422	gmc_cum_virt_mem_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to NM stack objects.
21423	gmc_cum_virt_mem_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to CM stack objects.
21424	gmc_cum_virt_mem_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to NM code objects.
21425	gmc_cum_virt_mem_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to CM code objects.
21426	gmc_cum_virt_mem_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to CM data objects.
21427	gmc_cum_virt_mem_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to file objects.
21428	gmc_cum_virt_mem_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to NM system library objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21429	gmc_cum_virt_mem_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to CM system library objects.
21430	gmc_cur_objects_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of temporary data objects.
21431	gmc_cur_objects_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of permanent data objects.
21432	gmc_cur_objects_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of NM stack objects.
21433	gmc_cur_objects_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of CM stack objects.
21434	gmc_cur_objects_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of NM code objects.
21435	gmc_cur_objects_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of CM code objects.
21436	gmc_cur_objects_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of CM data objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21437	gmc_cur_objects_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of file objects.
21438	gmc_cur_objects_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of NM system library objects.
21439	gmc_cur_objects_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of CM system library objects.
21440	gmc_cum_objects_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of temporary data objects.
21441	gmc_cum_objects_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of permanent data objects.
21442	gmc_cum_objects_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of NM stack objects.
21443	gmc_cum_objects_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of CM stack objects.
21444	gmc_cum_objects_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of NM code objects.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21445	gmc_cum_objects_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of CM code objects.
21446	gmc_cum_objects_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of CM data objects.
21447	gmc_cum_objects_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of file objects.
21448	gmc_cum_objects_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of NM system library objects.
21449	gmc_cum_objects_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of CM system library objects.
21450	gmc_ppf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the VPN cache was locked to service a primary page fault. A primary page fault is the first page fault.
21451	gmc_spf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the VPN cache was locked to service a secondary or greater page fault. A secondary page fault are those page faults that occur when trying to service the first (primary) page fault. In order to service the secondary page fault there could be a third level page fault and so on. All these page faults are denoted as secondary or greater page fault.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21452	gmc_npf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the VPN cache was locked for reasons other than to service page faults.
21453	gmc_num_npf_pages_locked (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages that were locked in the VPN cache for reasons other than page faults.
21454	gmc_dpf_for_ppf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page faults while doing the primary page fault locking.
21455	gmc_dpf_for_spf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page faults while doing the secondary page fault locking.
21456	gmc_dpf_for_npf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of data page fault when locking the VPN cache for reasons other than page faults.
21457	gmc_cpf_for_ppf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults while doing the primary page fault locking.
21458	gmc_cpf_for_spf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults while doing the secondary page fault locking

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21459	gmc_cpf_for_npf_lock_in_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of code page faults when locking the VPN cache for reasons other than page faults.
21460	gmc_max_pageflt_depth (I32) Counter Type: Simple Counter Unit of Measurement: Count The maximum level of page faults that happened in the system.
21461	gmc_ppf_avoid_locking_vs_od (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of primary page faults that avoid locking the VSOD.
21462	gmc_spf_avoid_locking_vs_od (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of secondary or greater page faults that avoid locking the VSOD.
21463	gmc_npf_avoid_locking_vs_od (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the VPN cache is locked for reasons other than page faults but avoids locking the VSOD.
21464	gmc_ppf_hits_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the virtual page for which a primary page fault is serviced is found in the VPN cache.
21465	gmc_spf_hits_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the virtual page for which a secondary page fault is serviced is found in the VPN cache.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21466	gmc_npf_hits_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the VPN cache is locked for reasons other than page faults and finds the virtual page in the VPN cache.
21467	gmc_ppf_misses_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the virtual page for which a primary page fault is serviced is not found in the VPN cache.
21468	gmc_spf_misses_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the virtual page for which a secondary page fault is serviced is not found in the VPN cache.
21469	gmc_npf_misses_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the VPN cache is locked for reasons other than page faults and does not find the virtual page in the VPN cache.
21470	gmc_ppf_overlay_recoveries_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for primary page fault overlay recoveries.
21471	gmc_spf_overlay_recoveries_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for secondary page fault overlay recoveries.
21472	gmc_npf_overlay_recoveries_vpn_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for non-page fault overlay recoveries.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21473	gmc_ics_lock_in_vpn_caches (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the VPN cache was locked from the ICS.
21474	gmc_pages_for_ics_lock_in_vpn_caches (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of pages in the VPN cache that were locked from the ICS.
21475	gmc_unlock_from_vpn_caches (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for calls to unlock from the VPN cache.
21476	gmc_pages_for_unlock_from_vpn_caches (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for number of pages unlocked in the VPN cache.
21477	gmc_flag_page_as_not_virgins (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for pages that are flagged as non-virgin.
21478	gmc_unneeded_flag_page_as_not_virgins (I32) Counter Type: Simple Counter Unit of Measurement: Count Number of calls for other VSM procedures to get information from the VPN cache.
21479	gmc_cached_vpn_to_secondary_addresses (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for the pages in the VPN cache that have a set of information returned to them.
21480	gmc_cached_vpn_lists (I32) Counter Type: Simple Counter Unit of Measurement: Count Counter for list of all the contiguous VPN ranges that are in the VPN cache within the specified VPN range.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21481	gmc_cached_vs_ll_headers (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of locality list headers.
21482	gmc_lock_vpn_caches (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the VPN cache was locked.
21483	gmc_extent_faults_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count Counter for number of extent faults.
21549	gmc_sl_startg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of system transactions started.
21550	gmc_sl_nested_startg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of nested system start transactions.
21551	gmc_sl_endg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of non-nested system transactions ended.
21552	gmc_sl_abortg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of non-nested system transactions aborted.
21553	gmc_sl_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented on each system log post.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21554	gmt_sl_total_between_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the total time spent between system log posts.
21555	gmt_sl_max_between_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the longest time spent two system log posts.
21556	gmt_sl_time_log_write_ended (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the ending time of the last system log post.
21557	gmc_sl_blocked_all_caches_full (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time all cache blocks in revolving door are full.
21558	gmc_sl_total_blocked_pin_for_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of pins blocked by system log posts.
21559	gmc_sl_max_blocked_pin_per_log_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the maximum number of pins blocked by system log posts.
21560	gmc_sl_blocked_on_commit (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time the XM caller is blocked until his transaction is posted on disk.
21561	gmc_sl_over_commit_threshold (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time the system log is posted because there are too many pins.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21562	gmc_sl_timer_pops (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time a timer pop forces a system log post.
21563	gmc_sl_full_buffer (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time a full log buffer forces a system log post.
21564	gmc_sl_full_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time a full cache block forces a system log post.
21565	gmc_sl_over_page_threshold (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time system log is posted because there are too many frozen pages.
21566	gmc_sl_num_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of system checkpoints.
21567	gmt_sl_total_between_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This counter tracks the total time spent not doing system checkpoints.
21568	gmt_sl_min_between_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This counter tracks the shortest time spent between two system checkpoints.
21569	gmt_sl_max_between_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This counter tracks the longest time spent between two system checkpoints.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21570	gmt_sl_total_duration_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the time spent doing system checkpoints.
21571	gmt_sl_min_duration_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the shortest time spent doing a system checkpoint.
21572	gmt_sl_max_duration_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the longest system checkpoint.
21573	gmt_sl_begin_time_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the time that the last system checkpoint began.
21574	gmc_sl_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of checkpoints blocked because previous system checkpoint hasn't completed.
21575	gmt_sl_total_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the total time that system checkpoints have been blocked waiting for other checkpoints to complete.
21576	gmt_sl_max_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the longest wait that one checkpoint has had for the previous system checkpoint to complete.
21577	gmt_sl_min_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter track the shortest wait that one checkpoint has had for the previous system checkpoint to complete.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21578	gmc_sl_freeze_cause_post (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that frozen pages cause a post.
21579	gmc_sl_overwrite_freeze (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that pages in XMs frozen page cache have been updated while still frozen.
21580	gmc_sl_pgs_frozen_cum (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the cumulative number of pages frozen by XM.
21581	gmc_sl_pgs_unfrozen_cum (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the cumulative number of pages unfrozen by XM.
21582	gmc_sl_froze_0_3_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 0 to 3 frozen pages were unfrozen and deleted from the XM cache.
21583	gmc_sl_froze_4_7_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 4 to 7 frozen pages were unfrozen and deleted from the XM cache.
21584	gmc_sl_froze_8_15_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 8 to 15 frozen pages were unfrozen and deleted from the XM cache.
21585	gmc_sl_froze_16_23_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 16 to 23 frozen pages were unfrozen and deleted from the XM cache.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21586	gmc_sl_froze_24_31_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 24 to 31 frozen pages were unfrozen and deleted from the XM cache.
21587	gmc_sl_froze_32_47_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 32 to 47 frozen pages were unfrozen and deleted from the XM cache.
21588	gmc_sl_froze_48_63_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 48 to 63 frozen pages were unfrozen and deleted from the XM cache.
21589	gmc_sl_froze_64_95_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 64 to 95 frozen pages were unfrozen and deleted from the XM cache.
21590	gmc_sl_froze_96_127_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 96 to 127 frozen pages were unfrozen and deleted from the XM cache.
21591	gmc_sl_froze_128_more_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 128 or more frozen pages were unfrozen and deleted from the XM cache.
21592	gmc_sl_post_log_1pg (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 1 page was posted to the system log.
21593	gmc_sl_post_log_2pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 2 pages were posted to the system log.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21594	gmc_sl_post_log_3_4pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 3 to 4 pages were posted to the system log.
21595	gmc_sl_post_log_5_8pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 5 to 8 pages were posted to the system log.
21596	gmc_sl_post_log_9_12pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 9 to 12 pages were posted to the system log.
21597	gmc_sl_post_log_13_16pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 13 to 16 pages were posted to the system log.
21598	gmc_sl_post_log_17_20pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 17 to 20 pages were posted to the system log.
21599	gmc_sl_post_log_21_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 21 pages were posted to the system log.
21600	gmc_ul_startg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of user transactions started.
21601	gmc_ul_nested_startg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of nested user start transactions. gmc_extent_faults_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count Counter for number of extent faults.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21602	gmc_ul_endg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of non-nested user transactions ended.
21603	gmc_ul_abortg_trans (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of non-nested user transactions aborted.
21604	gmc_ul_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented on each user log post.
21605	gmt_ul_total_between_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the total time spent between user log posts.
21606	gmt_ul_max_between_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the longest time spent between two user log posts.
21607	gmt_ul_time_log_write_ended (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the ending time of the last user log post.
21608	gmc_ul_blocked_all_caches_full (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time all cache blocks in revolving door are full.
21609	gmc_ul_total_blocked_pin_for_log_writes (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of pins blocked by user log posts.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21610	gmc_ul_max_blocked_pin_per_log_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the maximum number of pins blocked by user log posts.
21611	gmc_ul_blocked_on_commit (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time the XM caller is blocked until his transaction is posted on disk.
21612	gmc_ul_over_commit_threshold (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time the user log is posted because there are too many pins.
21613	gmc_ul_timer_pops (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time a timer pop forces a user log post.
21614	gmc_ul_full_buffer (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time a full log buffer forces a user log post.
21615	gmc_ul_full_cache (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time a full cache block forces a user log post.
21616	gmc_ul_over_page_threshold (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is incremented each time user log is posted because there are too many frozen pages.
21617	gmc_ul_num_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the number of user checkpoints.
21618	gmt_ul_total_between_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This counter tracks the total time spent not doing user checkpoints.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21619	gmt_ul_min_between_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the shortest time spent between two user checkpoints.
21620	gmt_ul_max_between_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the longest time spent between two user checkpoints.
21621	gmt_ul_total_duration_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the time spent doing user checkpoints.
21622	gmt_ul_min_duration_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the shortest time spent doing a user checkpoint.
21623	gmt_ul_max_duration_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the longest user checkpoint.
21624	gmt_ul_begin_time_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the time that the last user checkpoint began.
21625	gmc_ul_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of checkpoints blocked because previous user checkpoint hasn't completed.
21626	gmt_ul_total_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time This counter tracks the total time that user checkpoints have been blocked waiting for other checkpoints to complete.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21627	gmt_ul_max_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This counter tracks the longest wait that one checkpoint has had for the previous user checkpoint to complete.
21628	gmt_ul_min_blocked_checkpoints (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This counter track the shortest wait that one checkpoint has had for the previous user checkpoint to complete.
21629	gmc_ul_freeze_cause_post (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that frozen pages cause a post.
21630	gmc_ul_overwrite_freeze (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that pages in XMs frozen page cache have been updated while still frozen.
21631	gmc_ul_pgs_frozen_cum (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the cumulative number of pages frozen by XM.
21632	gmc_ul_pgs_unfrozen_cum (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter counts the cumulative number of pages unfrozen by XM.
21633	gmc_ul_froze_0_3_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 0 to 3 frozen pages were unfrozen and deleted from the XM cache.
21634	gmc_ul_froze_4_7_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 4 to 7 frozen pages were unfrozen and deleted from the XM cache.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21635	gmc_ul_froze_8_15_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 8 to 15 frozen pages were unfrozen and deleted from the XM cache.
21636	gmc_ul_froze_16_23_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 16 to 23 frozen pages were unfrozen and deleted from the XM cache.
21637	gmc_ul_froze_24_31_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 24 to 31 frozen pages were unfrozen and deleted from the XM cache.
21638	gmc_ul_froze_32_47_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 32 to 47 frozen pages were unfrozen and deleted from the XM cache.
21639	gmc_ul_froze_48_63_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 48 to 63 frozen pages were unfrozen and deleted from the XM cache.
21640	gmc_ul_froze_64_95_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 64 to 95 frozen pages were unfrozen and deleted from the XM cache.
21641	gmc_ul_froze_96_127_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 96 to 127 frozen pages were unfrozen and deleted from the XM cache.
21642	gmc_ul_froze_128_more_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 128 or more frozen pages were unfrozen and deleted from the XM cache.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21643	gmc_ul_post_log_1pg (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 1 page was posted to the user log.
21644	gmc_ul_post_log_2pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 2 pages were posted to the user log.
21645	gmc_ul_post_log_3_4pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 3 to 4 pages were posted to the user log.
21646	gmc_ul_post_log_5_8pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 5 to 8 pages were posted to the user log.
21647	gmc_ul_post_log_9_12pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 9 to 12 pages were posted to the user log.
21648	gmc_ul_post_log_13_16pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 13 to 16 pages were posted to the user log.
21649	gmc_ul_post_log_17_20pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 17 to 20 pages were posted to the user log.
21650	gmc_ul_post_log_21_pgs (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter tracks the number of times that 21 pages were posted to the user log.

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21651	<p>gmc_cur_blk_sql_latch_level_5_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 5 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21652	<p>gmc_cur_blk_sql_latch_level_6_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 6 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21653	<p>gmc_cur_blk_sql_latch_level_7_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 7 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21654	<p>gmc_cur_blk_sql_latch_level_8_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 8 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21655	<p>gmc_cur_blk_sql_latch_level_9_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 9 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21656	<p>gmc_cur_blk_sql_latch_level_10_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 10 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21657	<p>gmc_cur_blk_sql_latch_level_11_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 11 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21658	<p>gmc_cur_blk_sql_latch_level_12_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 12 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21659	<p>gmc_cur_blk_sql_latch_level_13_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 13 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21660	<p>gmc_cur_blk_sql_latch_level_14_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 14 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21661	<p>gmc_cur_blk_sql_latch_level_15_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 15 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21662	<p>gmc_cur_blk_sql_latch_level_16_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 16 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21663	<p>gmc_cur_blk_sql_latch_level_17_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 17 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21664	<p>gmc_cur_blk_sql_latch_level_18_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 18 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21665	<p>gmc_cur_blk_sql_latch_level_19_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 19 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21666	<p>gmc_cur_blk_sql_latch_level_20_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 20 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21667	<p>gmc_cur_blk_sql_latch_level_21_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 21 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21668	<p>gmc_cur_blk_sql_latch_level_22_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 22 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21669	<p>gmc_cur_blk_sql_latch_level_23_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 23 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21670	<p>gmc_cur_blk_sql_latch_level_24_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 24 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21671	<p>gmc_cur_blk_sql_latch_level_25_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 25 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21672	<p>gmc_cur_blk_sql_latch_level_26_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 26 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21673	<p>gmc_cur_blk_sql_latch_level_27_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 27 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21674	<p>gmc_cur_blk_sql_latch_level_28_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 28 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21675	<p>gmc_cur_blk_sql_latch_level_29_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 29 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21676	<p>gmc_cur_blk_sql_latch_level_30_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 30 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21677	<p>gmc_cur_blk_sql_latch_level_31_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 31 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
21678	<p>gmc_cur_blk_sql_latch_level_32_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The current number of processes that are blocked on a level 32 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table E-1. Global Counter Information (continued)

Num	Counter Name (Data Type) and Description
21679	gmc_cur_blk_release (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of times that a process has given up the CPU during process termination. The usual value of the counter is zero.
21680	gmc_cur_blk_memmgt_pseudo_ioread (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of times that a process has stopped with a block_and_send event descriptor and is initiating a read.
21681	gmc_cur_blk_memmgt_pseudo_iowrite (I32) Counter Type: Simple Counter Unit of Measurement: Count The current number of times that a process has stopped with a block_and_send event descriptor and is initiating a write.

Process Counters

This appendix provides descriptions of the item numbers for the counters that measure process-specific events as well as descriptions of the events measured. These item numbers are valid for `MIPROCON` and `MIPROCGET` calls.

Note

Counters associated with TurboIMAGE can be enabled, but they are not incremented.

Table F-1. Process Counter Information

Num	Counter Name (Data Type) and Description
25000	pmc_tot_buffer_access (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE buffer accesses done by the process.
25001	pmc_buffer_overlay (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a TurboIMAGE buffer is overlaid by the process.
25002	pmc_buffer_hits (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process finds the required buffer in the TurboIMAGE buffer pool.
25003	pmc_buffer_full (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process does not find the required buffer in the TurboIMAGE buffer pool.
25004	pmc_buffer_cnt_change (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
25005	pmc_buffer_pri_readj (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
25006	pmc_find_on_putdelete_q (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process calls DBFIND and waits on the DBPUT or DBDELETE queue.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25007	pmc_blocks_ser_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process waits on TurboIMAGE block reads for serial read.
25008	pmc_blocks_cal_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process waits on TurboIMAGE block reads for calculated read.
25009	pmc_blocks_put_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE Blocks read by the process for DBput master.
25010	pmc_write_put_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_write calls by the process for DBPUT detail.
25011	pmc_write_del_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_write calls by the process for DBDELETE detail.
25012	pmc_tot_put_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBPUT details by the process.
25013	pmc_tot_del_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBDELETE details by the process.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25014	pmc_tot_ser_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for serial read by the process.
25015	pmc_tot_cal_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read for calculated read by the process.
25016	pmc_tot_put_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of DBPUT masters for the process.
25017	pmc_prim_hash_collision (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of primary hash collisions encountered by the process.
25018	pmc_dirt_buffer_overlay (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process overlays the dirty TurboIMAGE buffer (for single user only).
25019	pmc_pred_lock_collision (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of predicate lock collisions (for \geq , \leq , $>$ and $<$) encountered by the process.
25020	pmc_blocks_chain_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read by the process for chain read.
25021	pmc_write_put_master (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disc_sm_start_wrtie calls by the process for DBPUT master.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25022	<p>pmc_write_del_master (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of disc_sm_start_wrtie calls by the process for DBdelete Master.</p>
25023	<p>pmc_wlabel_put_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>Not instrumented.</p>
25024	<p>pmc_wlabel_del_detail (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>Not Instrumented.</p>
25025	<p>pmc_tot_del_master (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of DBDELETE master by the process.</p>
25026	<p>pmc_tot_det_paths (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>Not instrumented.</p>
25027	<p>pmc_tot_dbbegin (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process called DBBEGIN.</p>
25028	<p>pmc_tot_dbcontrol (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process called DBCONTROL.</p>
25029	<p>pmc_tot_dbend (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process called DBEND.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25030	pmc_tot_dbfind (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBFIND.
25031	pmc_tot_dbget (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBGET.
25032	pmc_tot_DBinfo (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBINFO.
25033	pmc_tot_dbblock (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBLOCK.
25034	pmc_tot_dbopen (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBOPEN.
25035	pmc_tot_dbunlock (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBUNLOCK.
25036	pmc_tot_dbupdate (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBUPDATE.
25037	pmc_tot_chain_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of TurboIMAGE blocks read by the process for chain read.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25038	pmc_tot_dbput (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBPUT.
25039	pmc_tot_dbdelete (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBDELETE.
25040	pmc_pri_demote_3to2 (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
25041	pmc_pri_demote_2to1 (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
25042	pmc_pri_demote_1to0 (I32) Counter Type: Simple Counter Unit of Measurement: Count Not instrumented.
25043	pmc_tot_dbclose (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBCLOSE.
25044	pmc_tot_dberror (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBERROR.
25045	pmc_tot_dbexplain (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBEXPLAIN.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25046	pmc_tot_dbmemo (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the process called DBMEMO.
25047	pmt_process_time (I32) Counter Type: Simple Counter Unit of Measurement: ML Time The total CPU time used by a process.
25048	pmt_ready_queue (I32) Counter Type: Simple Counter Unit of Measurement: ML Time The total time spent by a process in the ready queue waiting to be served by the CPU.
25049	pmc_launch (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a process is launched (acquires the CPU).
25050	pmc_process_priority (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a process's priority is adjusted.
25051	pmc_stop_nm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked for an I/O to complete on a native mode code page fault.
25052	pmc_stop_nm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked for an I/O to complete on a native mode stack page fault.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25053	<p>pmc_stop_nm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an I/O to complete on a native mode transient page fault (heap, swappable table).</p>
25054	<p>pmc_stop_file_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an I/O to complete on a file page fault.</p>
25055	<p>pmc_stop_cm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an I/O to complete on a compatible mode code page fault.</p>
25056	<p>pmc_stop_cm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an I/O to complete on a compatible mode stack page fault.</p>
25057	<p>pmc_stop_cm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an I/O to complete on a compatibility mode transient page fault.</p>
25058	<p>pmc_stop_term_read (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for terminal read.</p>
25059	<p>pmc_stop_term_write (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process waited for a terminal write, such as console I/O or gen message for WARN and TELL messages.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25060	<p>pmc_stop_disc_io (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an I/O to complete on a compatibility mode transient page fault.</p>
25061	<p>pmc_stop_other_io (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for non-disk I/O devices.</p>
25062	<p>pmc_stop_ipc_trans_complete (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a process blocked on a port with transaction completed as one of their options (for example, the IOWAIT intrinsic).</p>
25063	<p>pmc_stop_sir_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for an SIR.</p>
25064	<p>pmc_stop_rin_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for a RIN.</p>
25065	<p>pmc_stop_mem_mgr_prefetch (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for a memory manager prefetch.</p>
25066	<p>pmc_stop_quantum_expiration (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process used up it's time quantum.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25067	<p>pmc_stop_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for a timeout or pause with one second or less.</p>
25068	<p>pmc_stop_parent_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process was waiting for its parents to wake it up.</p>
25069	<p>pmc_stop_cntl_block_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for a control block on a semaphore.</p>
25070	<p>pmc_stop_child_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for its children to wake it up.</p>
25071	<p>pmc_stop_data_comm_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for data communication.</p>
25072	<p>pmc_stop_rit_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for a RIT (operator reply).</p>
25073	<p>pmc_stop_disp_work (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process was preempted by the dispatcher to work on higher priority system processes (power failure, grey page cleanup, replenish critical pool, fetch I/O, or system fetch).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25074	pmc_stop_port_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked on a port (default IPC wait).
25075	pmc_stop_mail_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked on a port for a MAIL (old type of IPC interface that existed prior to message file implementation).
25076	pmc_stop_junk_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked on a port for a JUNK wait (don't care type wait).
25077	pmc_stop_message_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked on a port for a MESSAGE (basic IPC message file).
25078	pmc_stop_impede (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process impeded (mostly used by the file system for synchronization purpose).
25079	pmc_stop_break_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process was in break mode.
25080	pmc_stop_wait_queue (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked to be put on a wait queue by ports. (Table management running out of entries waits for additional spaces to be allocated.)

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25081	<p>pmc_stop_memmgt_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because the memory manager waits for proper I/O synchronization, excluding user I/O request such as POST.</p>
25082	<p>pmc_stop_port_blocked_make_present (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked until the requested port is present.</p>
25083	<p>pmc_stop_file_blocked (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked on a port for posting pages through the call CM_POST.</p>
25084	<p>pmc_stop_file_unblocked (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This block on port not used. This counter should be zero.</p>
25085	<p>pmc_stop_storage_mgt (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process blocked on a port through storage management.</p>
25086	<p>pmc_stop_user_to_debug_msg (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked due to breakpoint contention.</p>
25087	<p>pmc_stop_io_config_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because devices are being configured or released.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25088	<p>pmc_stop_pfp_reply_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because the port facility process needs to be created, initialized, or checked.</p>
25089	<p>pmc_stop_db_monitor_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because the database monitor (SQL) waits for the DB CLEAN_UP process to finish cleaning up the aborted DB processes before closing the database.</p>
25090	<p>pmc_stop_fill_disc_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because the new file extent in secondary storage needs to be initialized with fill characters for all virgin pages.</p>
25091	<p>pmc_stop_hlio_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because HLIO aborts the I/O.</p>
25092	<p>pmc_stop_fs_tio_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked because Terminal I/O (TIO) fast write in DTS (BND LDM) waits for a reply message from the device manager or the buffer management.</p>
25093	<p>pmc_stop_mem_mgr_post_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for I/O completion when explicitly posting pages to disk.</p>
25094	<p>pmc_stop_signal_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked for a delay or timer on a standard signal port.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25095	<p>pmc_stop_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process got preempted by a higher-priority process due to process awakening (IPC wait other than disk I/O completion).</p>
25096	<p>pmc_stop_disc_io_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process got preempted by a higher-priority process due to disk I/O completion. This includes page fault, post, and prefetch.</p>
25097	<p>pmc_stop_priority_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process got preempted by a higher-priority process due to priority boosting or dropping.</p>
25098	<p>pmc_stop_sql_lock_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked to acquire an SQL lock. This lock is required for user data (a tuple, a page, or a relation).</p>
25099	<p>pmc_stop_sql_latch_level_1_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 1 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25100	<p>pmc_stop_sql_latch_level_2_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 2 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25101	<p>pmc_stop_sql_latch_level_3_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 3 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25102	<p>pmc_stop_sql_latch_level_4_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 4 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25103	<p>pmc_stop_sql_buffer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocked to change the state of a page in the SQL buffer. The buffer is used to hold a page (4 K) of user data. The page can be in a number of states (being updated, in transit in, in transit out). When a process requests a page be placed in a state that conflicts with its current state, the process blocks.</p>
25104	<p>pmc_stop_long_pause_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process paused for 2 seconds or more.</p>
25105	<p>pmc_stop_mem_mgr_freeze_and_other (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process blocks on freeze and corner cases other than page_fault, prefetch, and freeze. This counter is predominantly block on freeze.</p>
25106	<p>pmc_stop_other_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks for other block reasons, such as giving up CPU because the process wasn't removed accordingly. This includes all reasons other than those that are measured.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25107	pmt_stop_nm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a native mode code page fault.
25108	pmt_stop_nm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a native mode stack page fault.
25109	pmt_stop_nm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a native mode transient page fault (heap, swappable table).
25110	pmt_stop_file_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a file page fault.
25111	pmt_stop_cm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a compatible mode code page fault.
25112	pmt_stop_cm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a compatible mode stack page fault.
25113	pmt_stop_cm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for an I/O to complete on a compatibility mode transient page fault.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25114	pmt_stop_term_read (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for terminal read.
25115	pmt_stop_term_write (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for terminal write.
25116	pmt_stop_disc_io (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for disk I/O.
25117	pmt_stop_other_io (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for non disk I/O.
25118	pmt_stop_ipc_trans_complete (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked on a port with transaction completed as one of their options(for example, IOWAIT).
25119	pmt_stop_sir_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for an SIR.
25120	pmt_stop_rin_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for a RIN.
25121	pmt_stop_mem_mgr_prefetch (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for a prefetch.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25122	pmt_stop_quantum_expiration (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for quantum expiration. The value of this counter should be zero since the process is not removed from the ready queue.
25123	pmt_stop_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for a timeout or pause with one second or less.
25124	pmt_stop_parent_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked until its parent awakened this process.
25125	pmt_stop_cntl_block_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for a control block on a semaphore.
25126	pmt_stop_child_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked until its child awakened this process.
25127	pmt_stop_data_comm_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for data communication.
25128	pmt_stop_rit_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for a RIT.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25129	pmt_stop_disp_work (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked due to preemption by the dispatcher to work on a higher-priority process. The value should be zero since the process is not removed from the ready queue.
25130	pmt_stop_port_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked on a port (default IPC wait).
25131	pmt_stop_mail_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked on a port for a MAIL (old type of IPC interface that existed prior to message file implementation).
25132	pmt_stop_junk_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked on a port for a JUNK wait (don't care type wait).
25133	pmt_stop_message_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked on a port for a MESSAGE (basic IPC message file).
25134	pmt_stop_impede (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process impeded (mostly used by the file system for synchronization purpose).
25135	pmt_stop_break_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process was in break mode.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25136	pmt_stop_wait_queue (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked to be put on a wait queue by PORTS (table management running out of entries waits for additional spaces to be allocated).
25137	pmt_stop_memmgt_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked because the memory manager waits for proper I/O synchronization, excluding user I/O requests such as POST.
25138	pmt_stop_port_blocked_make_present (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked until the requested port is present.
25139	pmt_stop_file_blocked (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked for posting pages through the call CM_POST.
25140	pmt_stop_file_unblocked (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time This block on port not used. This counter should be zero.
25141	pmt_stop_storage_mgt (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked on a port through storage management.
25142	pmt_stop_user_to_debug_msg (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that the process blocked due to breakpoint contention.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25143	pmt_stop_io_config_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked because devices are being configured or released.
25144	pmt_stop_pfp_reply_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked because the port facility process needs to be created, initialized, or checked.
25145	pmt_stop_db_monitor_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked because the database monitor (SQL) waits for the DB CLEAN_UP process to finish cleaning up the aborted DB processes before closing the database.
25146	pmt_stop_fill_disc_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked because the new file extent in secondary storage need to be initialized with fill characters for all virgin pages.
25147	pmt_stop_hlio_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked because HLIO aborts the I/O.
25148	pmt_stop_fs_tio_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked because Terminal I/O (TIO) fast write in DTS (BND LDM) waits for a reply message from the device manager or the buffer management.
25149	pmt_stop_mem_mgr_post_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total length of time that the process blocked for I/O completion when explicitly posting pages to disk.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25150	<p>pmt_stop_signal_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked for a delay or timer on a standard signal port.</p>
25151	<p>pmt_stop_preemption (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process was blocked due to preemption by a higher- priority process due to process awakening (IPC wait other than disk I/O completion). This value should be zero since the process is not removed from the ready queue.</p>
25152	<p>pmt_stop_disc_io_preemption (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process was blocked due to preemption by a higher priority process due to disk I/O completion. This includes page fault, post and prefetch. This value should be zero, since the process is not removed from the ready queue.</p>
25153	<p>pmt_stop_priority_preemption (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process got preempted by a higher priority process due to priority boosting or dropping. This value should be zero, since the process is not removed from the ready queue.</p>
25154	<p>pmt_stop_sql_lock_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked to acquire SQL lock. This lock is required for user data (a tuple, a page or a relation).</p>
25155	<p>pmt_stop_sql_latch_level_1_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 1 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25156	<p>pmt_stop_sql_latch_level_2_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 2 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25157	<p>pmt_stop_sql_latch_level_3_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 3 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25158	<p>pmt_stop_sql_latch_level_4_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 4 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25159	<p>pmt_stop_sql_buffer_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked to change the state of a page in the SQL buffer. The buffer is used to hold a page (4 K) of user data. The page can be in a number of states (for example, being updated, in transit in, in transit out). When a process requests a page be placed in a state that conflicts with its current state the process blocks.</p>
25160	<p>pmt_stop_long_pause_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a pause of 2 seconds or more.</p>
25161	<p>pmt_stop_mem_mgr_freeze_and_other (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on freeze and corner cases other than page_fault, prefetch, and freeze. This counter would be predominantly block on freeze.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25162	<p>pmt_stop_other_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocks for other block reasons such as giving up CPU because the process wasn't removed accordingly. This includes all reasons other than those that are measured.</p>
25163	<p>pmc_logical_termrds (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter counts the total number of terminal reads. One per read.</p>
25164	<p>pmc_logical_termwts (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter counts the total number of terminal writes. One per write.</p>
25165	<p>pmc_logical_rd_xfer_size (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter counts the total number of bytes of terminal reads.</p>
25166	<p>pmc_logical_wt_xfer_size (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter counts the total number of bytes of terminal writes.</p>
25167	<p>pmc_disc_read (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of disk reads by the process.</p>
25168	<p>pmc_disc_write (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of disk writes by the process.</p>
25169	<p>pmc_disc_read_size (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total transfer size of the disk reads.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25170	pmc_disc_write_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the disk writes.
25171	pmc_disc_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disk controls.
25172	pmc_terminal_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of terminal reads.
25173	pmc_terminal_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of terminal writes.
25174	pmc_terminal_read_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the terminal reads.
25175	pmc_terminal_write_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the terminal writes.
25176	pmc_terminal_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of terminal controls.
25177	pmc_tape_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of tape reads.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25178	pmc_tape_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of tape writes.
25179	pmc_tape_read_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the tape reads.
25180	pmc_tape_write_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the tape writes.
25181	pmc_tape_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of tape controls.
25182	pmc_printer_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes to printer.
25183	pmc_printer_write_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the write to printer.
25184	pmc_printer_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of controls to printer.
25185	pmc_other_io_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of I/O reads (other).

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25186	pmc_other_io_read_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the I/O reads (other).
25187	pmc_other_io_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of I/O writes (other).
25188	pmc_other_io_write_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The total transfer size of the I/O write (other).
25189	pmc_other_io_other (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of I/O (other).
25190	pmt_response_time (I32) Counter Type: Simple Counter Unit of Measurement: ML Time Total response time or response time to prompt for the process. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process, waiting for more data (another cr or enter). To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total response time for that interval. The average response time for this process can be obtained by dividing this value by the number of transactions for this process during the same interval.
25191	pmt_time_to_first_response (I32) Counter Type: Simple Counter Unit of Measurement: ML Time The total time spent waiting for the first write to the terminal for the process. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. The time is measured within the system before the write is sent to the terminal. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total first response for that interval. The average first response for this process can be obtained by dividing this value by the number of transactions for this process during the same interval.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
<p>25192</p>	<p>pmt_think_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total think time for the process. The total think time is defined by the system as the time beginning when HLIO sends a read request to the terminal, waiting for data (cr, enter), until HLIO receives a reply. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total think time for that interval. The average think time for this pin can be obtained by dividing this value by the number of transactions for this pin during the same interval.</p>
<p>25193</p>	<p>pmt_timed_read_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total think time spent waiting for a timed read expiration for the process. The times read wait is defined by the system as the time beginning when HLIO sends a read request to the terminal, waiting for data (cr, enter), until a timer pop completes the read. The user did not enter any data. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total time spent waiting for that interval. The average read wait for this process can be obtained by dividing this value by the number of timed reads for this process during the same interval. This time can also be expressed to show the percent of the think time that can be attributed to timed reads.</p>
<p>25194</p>	<p>pmc_timed_reads (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of reads completed as a result of timer pop for the process. Timed read wait is defined by the system as the time beginning when HLIO sends a read request to the terminal, waiting for data (cr, enter), until a timer pop completes the read. The user did not enter any data. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total timed reads that occurred during the interval. This count can be used to calculate the average read wait time during the interval.</p>
<p>25195</p>	<p>pmc_term_reads (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of reads completed as a result of user input for the process. Term read is defined by the system as a read that completes as a result of user input (cr, enter) and the think time of the read is between the thresholds. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total term reads that occurred during the interval. This count can be used to calculate the average response time, think time, and time to first response.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25196	<p>pmc_sw2nm_count (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process switched from Compatibility mode (CM) to Native mode (NM) and back.</p>
25197	<p>pmc_sw2cm_count (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process switched from Native mode (NM) to Compatibility mode (CM) and back.</p>
25198	<p>pmc_sw2cm_wrapdst (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that a process maps a CM data segment on top of an NM object so that the specified portion of the object can be accessed using data segment access methods. This is done usually for split stacks.</p>
25199	<p>pmt_cm_time (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total time spent by a process in Compatibility mode (CM).</p>
25200	<p>pmc_cum_alloc_main_mem (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of main memory pages allocated to the process.</p>
25201	<p>pmc_cur_main_mem_pg_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count</p> <p>The current number of main memory pages allocated to the process.</p>
25202	<p>pmc_pg_fault_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count</p> <p>The total number of page faults caused by the process.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25213	<p>pmc_cum_sec_storage_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to transient data objects for the process.</p>
25214	<p>pmc_cum_sec_storage_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to permanent data objects for the process.</p>
25215	<p>pmc_cum_sec_storage_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to NM stack objects for the process.</p>
25216	<p>pmc_cum_sec_storage_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to CM stack objects for the process.</p>
25217	<p>pmc_cum_sec_storage_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to NM code objects for the process.</p>
25218	<p>pmc_cum_sec_storage_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to CM code objects for the process.</p>
25219	<p>pmc_cum_sec_storage_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of secondary storage on disk allocated to CM data objects for the process.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25220	pmc_cum_sec_storage_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to file objects for the process.
25221	pmc_cum_sec_storage_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to NM system library objects for the process.
25222	pmc_cum_sec_storage_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of secondary storage on disk allocated to CM system library objects for the process.
25233	pmc_cum_virt_mem_temp_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to transient data objects for the process.
25234	pmc_cum_virt_mem_perm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to permanent data objects for the process.
25235	pmc_cum_virt_mem_nm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to NM stack objects for the process.
25236	pmc_cum_virt_mem_cm_stack (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to CM stack objects for the process.
25237	pmc_cum_virt_mem_nm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of bytes of virtual memory allocated to NM code objects for the process.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25238	<p>pmc_cum_virt_mem_cm_code (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of virtual memory allocated to CM code objects for the process.</p>
25239	<p>pmc_cum_virt_mem_cm_data (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of virtual memory allocated to CM data objects for the process.</p>
25240	<p>pmc_cum_virt_mem_file_object (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of virtual memory allocated to file objects for the process.</p>
25241	<p>pmc_cum_virt_mem_nm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of virtual memory allocated to NM system library objects for the process.</p>
25242	<p>pmc_cum_virt_mem_cm_sys_lib (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of bytes of virtual memory allocated to CM system library objects for the process.</p>
25243	<p>pmc_disc_read_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count</p> <p>The number of disk reads by object class by process.</p>
25244	<p>pmc_disc_write_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count</p> <p>The number of disk writes by object class by process.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25245	pmc_disc_read_size_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total transfer size of the disk read by object class by process.
25246	pmc_disc_write_size_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total transfer size of the disk write by object class by process.
25247	pmc_disc_other_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The total number of disk controls by object class by process.
25248	pmc_sm_read_count (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter records the number of disk file read requests made by a specific process.
25249	pmc_sm_write_count (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter records the number of disk file write requests made by a specific process.
25250	pmc_sm_read_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count This process counter tracks the number of disk file read requests that require that pages be brought into memory from disk before the read can complete.
25251	pmc_sm_r_soft_pf (I32) Counter Type: Simple Counter Unit of Measurement: Count This process counter tracks the number of disk file read requests that invoke an asynchronous fetch of disk pages to be brought into main memory.

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25252	<p>pmc_sm_write_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This process counter tracks the number of disk file write requests that require that pages be brought into memory from disk before the write can complete.</p>
25253	<p>pmc_sm_write_vp_miss (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This process counter tracks the number of disk file write requests that invoke allocation of a virgin page of memory.</p>
25254	<p>pmc_sm_w_soft_pf (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This process counter tracks the number of disk file read requests that invoke an asynchronous fetch of disk pages to be brought into main memory.</p>
25255	<p>pmc_sm_open_new_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of new disk files created by a process.</p>
25256	<p>pmc_sm_open_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk files opened by a process. Newly created files recorded in pmc_sm_open_new_file are included in this count. Note that for each FOPEN or HPFOPEN, this counter is incremented by four; once each for the directory root, account, group, and file.</p>
25257	<p>pmc_sm_close_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk files closed by a process. Note that for each FCLOSE, this counter is incremented by four; once each for the directory root, account, group, and file.</p>
25258	<p>pmc_sm_purge_file (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter records the number of disk files purged by a process.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25259	<p>pmc_ipc_transfer_to_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times the process is sent from a process environment to a port whose server can only be invoked on the ICS.</p>
25260	<p>pmc_pfp_msg_threshold_exceeded (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that the port facility process was invoked when the number of messages queued on a port exceeded the maximum threshold. This is to avoid the process that does a send to be stuck as the target stack for multiple invocations of the procedure server.</p>
25261	<p>pmc_semcl_que_1_3 (I32A) Counter Type: Semaphore Class Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks on a semaphore and finds the queue length of that semaphore ≥ 1 and ≤ 3. Note that the counter measures this event for a semaphore class that consists of a collection of semaphores.</p>
25262	<p>pmc_semcl_que_4_6 (I32A) Counter Type: Semaphore Class Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks on a semaphore and finds the queue length of that semaphore ≥ 4 and ≤ 6. Note that the counter measures this event for a semaphore class that consists of a collection of semaphores.</p>
25263	<p>pmc_semcl_que_7_over (I32A) Counter Type: Semaphore Class Counter Unit of Measurement: Count</p> <p>The total number of times that the process blocks on a semaphore and finds the queue length of that semaphore ≥ 7. Note that the counter measures this event for a semaphore class that consists of a collection of semaphores.</p>
25264	<p>pmc_stop_sql_latch_level_5_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 5 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25265	<p>pmc_stop_sql_latch_level_6_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 6 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25266	<p>pmc_stop_sql_latch_level_7_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 7 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25267	<p>pmc_stop_sql_latch_level_8_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 8 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25268	<p>pmc_stop_sql_latch_level_9_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 9 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25269	<p>pmc_stop_sql_latch_level_10_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 10 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25270	<p>pmc_stop_sql_latch_level_11_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 11 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25271	<p>pmc_stop_sql_latch_level_12_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 12 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25272	<p>pmc_stop_sql_latch_level_13_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 13 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25273	<p>pmc_stop_sql_latch_level_14_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 14 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25274	<p>pmc_stop_sql_latch_level_15_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 15 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25275	<p>pmc_stop_sql_latch_level_16_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 16 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25276	<p>pmc_stop_sql_latch_level_17_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 17 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25277	<p>pmc_stop_sql_latch_level_18_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 18 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25278	<p>pmc_stop_sql_latch_level_19_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 19 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25279	<p>pmc_stop_sql_latch_level_20_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 20 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25280	<p>pmc_stop_sql_latch_level_21_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 21 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25281	<p>pmc_stop_sql_latch_level_22_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 22 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25282	<p>pmc_stop_sql_latch_level_23_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 23 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25283	<p>pmc_stop_sql_latch_level_24_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 24 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25284	<p>pmc_stop_sql_latch_level_25_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 25 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25285	<p>pmc_stop_sql_latch_level_26_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 26 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25286	<p>pmc_stop_sql_latch_level_27_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 27 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25287	<p>pmc_stop_sql_latch_level_28_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 28 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25288	<p>pmc_stop_sql_latch_level_29_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 29 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25289	<p>pmc_stop_sql_latch_level_30_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 30 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25290	<p>pmc_stop_sql_latch_level_31_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 31 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25291	<p>pmc_stop_sql_latch_level_32_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that the process blocked on a level 32 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25293	<p>pmt_stop_sql_latch_level_5_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML Time</p> <p>The total length of time that the process blocked on a level 5 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25294	<p>pmt_stop_sql_latch_level_6_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML Time</p> <p>The total length of time that the process blocked on a level 6 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25295	<p>pmt_stop_sql_latch_level_7_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 7 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25296	<p>pmt_stop_sql_latch_level_8_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 8 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25297	<p>pmt_stop_sql_latch_level_9_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 9 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25298	<p>pmt_stop_sql_latch_level_10_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 10 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25299	<p>pmt_stop_sql_latch_level_11_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 11 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25300	<p>pmt_stop_sql_latch_level_12_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 12 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25301	<p>pmt_stop_sql_latch_level_13_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 13 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25302	<p>pmt_stop_sql_latch_level_14_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 14 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25303	<p>pmt_stop_sql_latch_level_15_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 15 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25304	<p>pmt_stop_sql_latch_level_16_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 16 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25305	<p>pmt_stop_sql_latch_level_17_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 17 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25306	<p>pmt_stop_sql_latch_level_18_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 18 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25307	<p>pmt_stop_sql_latch_level_19_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 19 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25308	<p>pmt_stop_sql_latch_level_20_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 20 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25309	<p>pmt_stop_sql_latch_level_21_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 21 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25310	<p>pmt_stop_sql_latch_level_22_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 22 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25311	<p>pmt_stop_sql_latch_level_23_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 23 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25312	<p>pmt_stop_sql_latch_level_24_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 24 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25313	<p>pmt_stop_sql_latch_level_25_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 25 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25314	<p>pmt_stop_sql_latch_level_26_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 26 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25315	<p>pmt_stop_sql_latch_level_27_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The total length of time that the process blocked on a level 27 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25316	<p>pmt_stop_sql_latch_level_28_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 28 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25317	<p>pmt_stop_sql_latch_level_29_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 29 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25318	<p>pmt_stop_sql_latch_level_30_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 30 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25319	<p>pmt_stop_sql_latch_level_31_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 31 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25320	<p>pmt_stop_sql_latch_level_32_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that the process blocked on a level 32 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
25321	<p>pmt_stop_release (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time</p> <p>The total length of time that a process has stopped by giving up the CPU during process termination. The usual value of the counter is zero.</p>
25322	<p>pmc_stop_memmgt_psuedo_ioread (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a process has stopped with a block_and_send event descriptor and is initiating a read.</p>

Table F-1. Process Counter Information (continued)

Num	Counter Name (Data Type) and Description
25323	pmt_stop_memmgt_psuedo_iowrite (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times a process has stopped with a block_and_send event descriptor and is initiating a write.
25324	pmt_stop_memmgt_psuedo_ioread (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that a process has stopped with a block_and_send event descriptor and is initiating a read.
25325	pmt_stop_memmgt_psuedo_iowrite (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of time that a process has stopped with a block_and_send event descriptor and is initiating a write.

I/O Counters

This appendix provides descriptions of the item numbers for the counters that measure I/O-specific events as well as descriptions of the events measured. These item numbers are valid for `MII00N` and `MII0GET` calls.

Table G-1. I/O Counter Information

Num	Counter Name (Data Type) and Description
23000	imc_logical_rd (I32) Counter Type: Simple Counter Unit of Measurement: Count Not implemented.
23001	imc_logical_wt (I32) Counter Type: Simple Counter Unit of Measurement: Count Not implemented.
23002	imc_logical_rd_xfer_size (I32) Counter Type: Simple Counter Unit of Measurement: Count Not implemented.
23003	imc_logical_wt_xfer_size (I32) Counter Type: Simple Counter Unit of Measurement: Count Not implemented.
23004	imc_rd_non_mm_user (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk read requests.
23005	imc_wt_non_mm_user (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk write requests.
23006	imc_nm_trans_page_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23007	imc_orderedio_read (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23008	imc_orderedio_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23009	imc_swaplist_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disk read requests.
23010	imc_swaplist_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of disk write requests.
23011	imc_rd_min_swaplist (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23012	imc_wt_min_swaplist (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23013	imc_cm_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk compatibility mode read requests.
23014	imc_cm_blocked_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk compatibility mode blocked read requests.
23015	imc_cm_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk compatibility mode write requests.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23016	imc_cm_blocked_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk compatibility mode blocked write requests.
23017	imc_cm_mapped_read (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23018	imc_rd_cm_mapped_blocked (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23019	imc_cm_mapped_write (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23020	imc_wt_cm_mapped_blocked (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23021	imc_orderedio_processed (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23022	imc_completion (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk I/O completions.
23023	imt_wt_round_trip_time (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The round trip time for writes for all types of I/O.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23024	imt_rd_round_trip_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The round trip time for reads for all types of I/O.
23025	imc_q_depth_0 (I32) Counter Type: Simple Counter Unit of Measurement: Count This disk I/O queue depth of zero.
23026	imc_q_depth_1 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth of one.
23027	imc_q_depth_2 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth of two.
23028	imc_q_depth_to_4 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth between three and four.
23029	imc_q_depth_to_8 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth between five and eight.
23030	imc_q_depth_to_16 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth between nine and sixteen.
23031	imc_q_depth_to_32 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth between seventeen and thirty-two.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23032	imc_q_depth_over_32 (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O queue depth over thirty-two.
23033	imc_q_maximum (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O maximum observed queue depth.
23034	imc_q_current (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O current queue depth.
23035	imc_q_cumulative (I32) Counter Type: Simple Counter Unit of Measurement: Count The disk I/O cumulative queue depth.
23036	imc_read_length (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk CM bytes to read.
23037	imc_write_length (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of non-disk CM bytes to write.
23038	imc_freeze_try (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23039	imc_control (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of controls issued.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23040	imc_read_xfer_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of bytes transferred for read request for all types of I/O.
23041	imc_write_xfer_size (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of bytes transferred for write request for all types of I/O.
23042	imc_freeze_swap (I32) Counter Type: Simple Counter Unit of Measurement: Count This counter is not instrumented.
23043	imt_request_secs (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The time that all requests wait in the disk device manager queue waiting for service. The imt_q_delay_time counter should be used to calculate the queue delay time associated with disk I/Os.
23044	imt_q_delay_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The time that all disk I/O requests spend in the disk device manager queue.
23045	imt_service_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The service time that all disk I/O requests take to be completed.
23046	imc_completed_io (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of disk I/Os completed.
23047	imc_aborted_io (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of disk I/Os aborted.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23048	imc_completed_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of disk I/O reads completed.
23049	imc_completed_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of disk I/O writes completed.
23050	imc_read_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The number of disk I/O read requests by object class.
23051	imc_write_objcl_ (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The number of disk I/O write requests by object class.
23052	imc_obj_xfer_read (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The number of bytes to read in the disk I/O request.
23053	imc_obj_xfer_write (I32A) Counter Type: Object Class Counter Unit of Measurement: Count The number of bytes to write in the disk I/O request.
23054	imt_response_time (I32) Counter Type: Simple Counter Unit of Measurement: ML Time This counter is valid only for TERMINAL device types. Response time or response time to prompt. This counter is accumulated for each LDEV. Response time is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until a read request is sent to the user terminal from a process (waiting for more data (another cr or enter)). To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total first response for that interval. The average response time for this LDEV can be obtained by dividing this value by the number of transactions for this LDEV during the same interval.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
<p>23055</p>	<p>imt_time_to_first_response (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>This counter is valid only for TERMINAL device types. Time spent waiting for the first write to the terminal. Time to first response is defined by the system as the time beginning when HLIO receives data from the user terminal (cr, enter), until the first byte is written to the terminal. The time is measured within the system before the write is sent to the terminal. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total first response for that interval. The average first response for this LDEV can be obtained by dividing this value by the number of transactions for this LDEV during the same interval.</p>
<p>23056</p>	<p>imt_think_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>This counter is valid only for TERMINAL device types. The amount of user think time. Think time is defined by the system as the time beginning when HLIO sends a read request to the terminal, waiting for data (cr, enter), until HLIO receives a reply. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total think time for that interval. The average think time for this LDEV can be obtained by dividing this value by the number of transactions for this LDEV during the same interval.</p>
<p>23057</p>	<p>imt_timed_read_wait (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>This counter is valid only for TERMINAL device types. Think time spent waiting for a timed read expiration. Timed read wait is defined by the system as the time beginning when HLIO sends a read request to the terminal, waiting for data (cr, enter), until a timer pop completes the read. The user did not enter any data. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total time spent waiting for that interval. The average read wait for this LDEV can be obtained by dividing this value by the number of timed reads for this LDEV during the same interval. This time can also be expressed to show the percent of the think time that can be attributed to timed reads.</p>

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23058	<p>imc_timed_reads (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is valid only for TERMINAL device types. Number of reads completed as a result of timer pop. Timed read wait is defined by the system as the time beginning when HLIO sends a read request to the terminal, waiting for data (cr, enter), until a timer pop completes the read. The user did not enter any data. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total timed reads that occurred during the interval. This count can be used to calculate the average read wait time during the interval.</p>
23059	<p>imc_term_reads (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This counter is valid only for TERMINAL device types. Number of reads completed as a result of user input. Term read is defined by the system as a read that completes as a result of user input (cr, enter) and the think time of the read is between the thresholds. To obtain the value of this counter, it is necessary for the tool to take two samples and subtract the first from the last. Since this counter is cumulative, this difference will provide the total term reads that occurred during the interval. This count can be used to calculate the average response time, think time, and time to first response.</p>
23060	<p>imc_vdm_q_depth_0 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be zero.</p>
23061	<p>imc_vdm_q_depth_1 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be one.</p>
23062	<p>imc_vdm_q_depth_2 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be two.</p>

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23063	<p>imc_vdm_q_depth_4 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be between 3 and 4.</p>
23064	<p>imc_vdm_q_depth_8 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be between 5 and 8.</p>
23065	<p>imc_vdm_q_depth_16 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be between 9 and 16.</p>
23066	<p>imc_vdm_q_depth_32 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth to be between 17 and 32.</p>
23067	<p>imc_vdm_q_depth_over_32 (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times a mirrored disk I/O request comes in and finds the VDM queue depth over 32.</p>
23068	<p>imc_vdm_q_depth_maximum (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>This tracks the maximum VDM queue depth.</p>
23069	<p>imt_vdm_repair_start_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time</p> <p>The start time of the current repair.</p>

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23070	imt_vdm_repair_stop_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The stop time of the last repair.
23071	imc_vdm_repair_k_bytes (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of kilobytes repaired.
23072	imc_vdm_repair_mibs (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of MIBs processed in the current repair.
23073	imt_vdm_repair_cum_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total amount of time since the start of the repair.
23074	imc_vdm_repair_percent_completed (I32) Counter Type: Simple Counter Unit of Measurement: Count The percentage of the repair that has completed.
23075	imc_vdm_read_q_cumulative (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of reads that have been queued in the VDM port.
23076	imc_vdm_write_q_cumulative (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes that have been queued in the VDM port.
23077	imc_vdm_read_q_wait_under_25 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of reads that are queued in the VDM port less than 25 milliseconds.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23078	imc_vdm_read_q_wait_under_50 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of reads that are queued in the VDM port less than 50 milliseconds.
23079	imc_vdm_read_q_wait_under_100 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of reads that are queued in the VDM port less than 100 milliseconds.
23080	imc_vdm_read_q_wait_under_200 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of reads that are queued in the VDM port less than 200 milliseconds.
23081	imc_vdm_read_q_wait_200_and_over (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of reads that are queued in the VDM port greater than or equal to 200 milliseconds.
23082	imc_vdm_write_q_wait_under_25 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes that are queued in the VDM port less than 25 milliseconds.
23083	imc_vdm_write_q_wait_under_50 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes that are queued in the VDM port less than 50 milliseconds.
23084	imc_vdm_write_q_wait_under_100 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes that are queued in the VDM port less than 100 milliseconds.
23085	imc_vdm_write_q_wait_under_200 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes that are queued in the VDM port less than 200 milliseconds.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23086	imc_vdm_write_q_wait_200_and_over (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of writes that are queued in the VDM port for greater than or equal to 200 milliseconds.
23087	imt_vdm_read_cum_q_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total amount of time that read requests are queued in the VDM port.
23088	imt_vdm_write_cum_q_wait (I32) Counter Type: Simple Counter Unit of Measurement: ML_Time The total amount of time that write requests are queued in the VDM port.
23089	imc_vdm_read_service_under_25 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored reads that have completed with service times less than 25 milliseconds.
23090	imc_vdm_read_service_under_50 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored reads that have completed with service times less than 50 milliseconds.
23091	imc_vdm_read_service_under_100 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored reads that have completed with service times less than 100 milliseconds.
23092	imc_vdm_read_service_under_200 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored reads that have completed with service times less than 200 milliseconds.

Table G-1. I/O Counter Information (continued)

Num	Counter Name (Data Type) and Description
23093	imc_vdm_read_service_200_and_over (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored reads that have completed with service times greater than or equal to 200 milliseconds.
23094	imc_vdm_write_service_under_25 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored writes that have completed with service times less than 25 milliseconds.
23095	imc_vdm_write_service_under_50 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored writes that have completed with service times less than 50 milliseconds.
23096	imc_vdm_write_service_under_100 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored writes that have completed with service times less than 100 milliseconds.
23097	imc_vdm_write_service_under_200 (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored writes that have completed with service times less than 200 milliseconds.
23098	imc_vdm_write_service_200_and_over (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of mirrored writes that have completed with service times greater than or equal to 200 milliseconds.
23099	imt_vdm_read_cum_service (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time Total amount of service time accumulated due to mirrored reads.
23100	imt_vdm_write_cum_service (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time Total amount of service time accumulated due to mirrored writes.

Examples

This appendix provides two examples to illustrate the use of measurement interface architected interfaces.

Example 1 - testmiaif1

```

$standard_level 'HP_MODCAL'$
Program testmiaif1(input,output,parm,info);
{
    test program to show how aifs work for the mi(measurement interface)
    4/18/90 SFM

    Program flow
    enable AIF access
    get conversion factors
    enable some global mi counters
    pause
    initialize internal tables with cumulative mi numbers

    loop 40 times and display values
    pause
    get mi data
    compute deltas for interval
    compute some performance measures
    display some of the cpu utilization values
    display some of the mi counters
    save current mi cumulative data for next interval
}
Type

{ the following set is used to generate array indexes      }
mi_aif_type=(

gmt_paused,
gmt_idle,
gmt_mem_mgr_sys_fetch,
gmt_mem_mgr_swapin,
gmt_mem_mgr_tos_trap,
gmt_disp_act,
gmt_as_process,

```

```

gmt_bs_process,
gmt_cs_process,
gmt_ds_process,
gmt_es_process,
gmt_cm_time,
gmc_stop_nm_code_page_fault,
gmc_stop_nm_stack_page_fault,
gmc_stop_nm_trans_page_fault,
gmc_stop_cm_code_page_fault,
gmc_stop_cm_stack_page_fault,
gmc_stop_cm_trans_page_fault,
gmc_stop_priority_preemption,
gmc_mem_clock_cycles,
gmc_mem_pressure,
gmc_sw2nm_count,
gmc_sw2cm_count
);
const
    number_of_counters=ord(gmc_sw2cm_count)+1;

type
    constant_array_type=array[0..number_of_counters] of integer;
Const

{ list of constants that need to be passed to the aif code }
c_gmt_paused=21176;
c_gmt_idle=21177;
c_gmt_mem_mgr_sys_fetch=21178;
c_gmt_mem_mgr_swapin=21179;
c_gmt_mem_mgr_tos_trap=21180;
c_gmt_disp_act=21181;
c_gmt_as_process=21182;
c_gmt_bs_process=21183;
c_gmt_cs_process=21184;
c_gmt_ds_process=21185;
c_gmt_es_process=21186;
c_gmt_cm_time=21317;
c_gmc_stop_nm_code_page_fault=21187;
c_gmc_stop_nm_stack_page_fault=21188;
c_gmc_stop_nm_trans_page_fault=21189;
c_gmc_stop_cm_code_page_fault=21191;
c_gmc_stop_cm_stack_page_fault=21192;
c_gmc_stop_cm_trans_page_fault=21193;
c_gmc_stop_priority_preemption=21233;
c_gmc_mem_clock_cycles=21306;
c_gmc_mem_pressure=21307;
c_gmc_sw2nm_count=21313;
c_gmc_sw2cm_count=21314;

{ the following is a array of constants passed to the AIF code }
mi_aif_constant_array=constant_array_type[

```

H-2 Examples

```

c_gmt_paused,
c_gmt_idle,
c_gmt_mem_mgr_sys_fetch,
c_gmt_mem_mgr_swapin,
c_gmt_mem_mgr_tos_trap,
c_gmt_disp_act,
c_gmt_as_process,
c_gmt_bs_process,
c_gmt_cs_process,
c_gmt_ds_process,
c_gmt_es_process,
c_gmt_cm_time,
c_gmc_stop_nm_code_page_fault,
c_gmc_stop_nm_stack_page_fault,
c_gmc_stop_nm_trans_page_fault,
c_gmc_stop_cm_code_page_fault,
c_gmc_stop_cm_stack_page_fault,
c_gmc_stop_cm_trans_page_fault,
c_gmc_stop_priority_preemption,
c_gmc_mem_clock_cycles,
c_gmc_mem_pressure,
c_gmc_sw2nm_count,
c_gmc_sw2cm_count,
0];

```

Type

```
constant_names_type=array[0..number_of_counters-1] of string[55];
```

Const

```
{ array of counter names; useful for debugging }
```

```

constant_names_array=constant_names_type[
'gmt_paused',
'gmt_idle',
'gmt_mem_mgr_sys_fetch',
'gmt_mem_mgr_swapin',
'gmt_mem_mgr_tos_trap',
'gmt_disp_act',
'gmt_as_process',
'gmt_bs_process',
'gmt_cs_process',
'gmt_ds_process',
'gmt_es_process',
'gmt_cm_time',
'gmc_stop_nm_code_page_fault',
'gmc_stop_nm_stack_page_fault',
'gmc_stop_nm_trans_page_fault',
'gmc_stop_cm_code_page_fault',
'gmc_stop_cm_stack_page_fault',
'gmc_stop_cm_trans_page_fault',

```

```

'gmc_stop_priority_preemption           ',
'gmc_mem_clock_cycles                   ',
'gmc_mem_pressure                        ',
'gmc_sw2nm_count                         ',
'gmc_sw2cm_count                         '
];
Const
    small_object_class_map=1;
Type
    status_type=packed record
        case integer of
            0:(all:integer);
            1:(
                info:shortint;
                subsys:shortint
            );
        end;
    Display=string[255];
    config_item_array_type=array[0..2] of integer;
    config_item_ptr_array_type=array[0..2] of globalanyptr;
Var
    parm:shortint;
    info:string[255];
    userid:integer;
    aif_status:status_type;
    item_nums:constant_array_type;
    item_statuses:array[0..24] of integer;
    mapping:integer;
    mi_status:status_type;
    item_data:constant_array_type;
    old_item_data:constant_array_type;
    delta_item_data:constant_array_type;
    buffer_pointer:globalanyptr;
    wait_time:real;
    j,i:integer;
    delta_time:integer;
    delta_mi_time:integer;
    old_time:integer;
    current_time:integer;

    config_items:config_item_array_type;
    config_status:config_item_array_type;
    config_items_ptr:config_item_ptr_array_type;
    mi_ticks_per_milli:integer;
    mi_round_factor:integer;
    milli_to_mi_time:real;
    total_accounted:integer;
    total_process:integer;
    ics_mi_time:integer;
    j1:integer;

```

```

$sysintr 'aifintr.pub.sys'$
procedure aifaccesson;intrinsic;
procedure aifsyswideget;intrinsic;
procedure aifaccessoff;intrinsic;
procedure aifscget;intrinsic;
$sysintr 'miintr'$
procedure miglobon;intrinsic;
procedure miglobget;intrinsic;
$sysintr$
procedure getprivmode;intrinsic;
procedure getusermode;intrinsic;
procedure pause;intrinsic;
function timer:integer;intrinsic;
$PAGE$
$TITLE 'Write_status - writes status & subsystem error messages'$
Procedure Write_Status(Message:Display;
                        Status:status_type);
BEGIN
Writeln(Message,' Subsystem: ',Status.subsys:1,
         ' Info: ',Status.Info:1);
END;
begin
userid:=666;
mapping:=small_object_class_map;
item_nums:=mi_aif_constant_array;;
buffer_pointer:=addr(item_data);
wait_time:=parm;

{ get conversion factors for MI internal time to milliseconds }

config_items[0]:=3042; {mi aif rounding factor }
config_items[1]:=3043; {mi aif ticks per millisecond }
config_items[2]:=0;
config_items_ptr[0]:=addr(mi_round_factor);
config_items_ptr[1]:=addr(mi_ticks_per_milli);

getprivmode;
aifaccesson(aif_status,userid);
if(aif_status.all<>0 )then
  write_status(' can''t get going',aif_status)
else
  begin
  aifscget(aif_status,config_items,config_items_ptr,
          config_status);
  if(aif_status.all<>0) then
    begin
    write_status(' can''t get the tick factor data ',aif_status);
    halt(50);
    end;
  end;

{ compute conversion factor for milliseconds => mi internal time }

```

```

milli_to_mi_time:=mi_ticks_per_milli/mi_round_factor;

miglobon(mi_status,item_nums,item_statuses,mapping);
if(mi_status.all<>0) then
  write_status(' trouble enabling the mi ',mi_status)
else
  begin

    pause(wait_time);
    miglobget(mi_status,item_nums,buffer_pointer,item_statuses,
              sizeof(item_data),mapping);
    if(mi_status.all<>0) then
      begin
        write_status(' error from getting mi data ',mi_status);
        halt(95);
      end;
    old_item_data:=item_data;
    old_time:=timer;
    for j:=1 to 40 do
      begin
        pause(wait_time);
        miglobget(mi_status,item_nums,buffer_pointer,item_statuses,
                  sizeof(item_data),mapping);
        if(mi_status.all<>0) then
          begin
            write_status(' error from getting mi data ',mi_status);
            halt(100);
          end;
        current_time:=timer;
        delta_time:=current_time-old_time;
        delta_mi_time:=trunc((delta_time+1)*milli_to_mi_time);
        { convert milliseconds to mi internal format time }

        { compute deltas for the interval }

        for i:=ord(gmt_paused) to ord(gmc_sw2cm_count) do
          begin
            delta_item_data[i]:=item_data[i]-old_item_data[i];
          end;

        { compute ICS by first summing all allocated for time then      }
        { subtracting from interval                                     }
        { the MI does not have a timer for interrupt control stack (ICS)}

        total_accounted:=0;
        for j1:=ord(gmt_paused) to ord(gmt_es_process) do
          total_accounted:=total_accounted+delta_item_data[j1];
          ics_mi_time:=delta_mi_time-total_accounted;

        { sum all process time    }

```



```

        for j1:=ord(gmt_as_process) to ord(gmt_es_process) do
            total_process:=total_process+delta_item_data[j1];
        }
{ display some of the computed values }
    writeln('delta time ',delta_time);
    writeln(' paused ',
        100*(delta_item_data[ord(gmt_paused)]/delta_mi_time):4:2,
        ' idle ',
        100*(delta_item_data[ord(gmt_idle)]/delta_mi_time):4:2,
        ' ics  ',
        100*(ics_mi_time/delta_mi_time):4:2,
        ' cs  ',
        100*(delta_item_data[ord(gmt_cs_process)]/delta_mi_time):4:2,
        ' ds  ',
        100*(delta_item_data[ord(gmt_ds_process)]/delta_mi_time):4:2,
        ' es  ',
        100*(delta_item_data[ord(gmt_es_process)]/delta_mi_time):4:2,
        '' );

{ display some none zero counters from the MI }

    for i:=ord(gmc_stop_nm_code_page_fault) to ord(gmc_sw2cm_count) do
        begin
            if(delta_item_data[i]<>0) then
                writeln(constant_names_array[i]:20,
                    delta_item_data[i]:1);
            end;
        }

{ save current cumulative data for next delta calculation }

        old_item_data:=item_data;
        old_time:=current_time;
        end;
    end;
end;
aifaccessoff(aif_status,userid);
getusermode;
end.

```

Example 2 - testmiaif2

```
$standard_level 'HP_MODCAL'$
$ list on $
$ statement_number on $
$ code_offsets on $
$ tables on $
{$xref on $ }
Program testmiaif2(input,output,param,info);

{*****}
{ test program to show how aifs work for the mi(measurement interface) }
{ the program look at disks reads/writes and compute them. }
{ The program all display object class read count per object. }
{ }
{ }
{ the algorithm: }
{ getprivmode }
{ turn on I/O portions of the MI with appropriate counter numbers }
{ get first measurement (to set base values ) }
{ get measurement }
{ for 1 to max_ldevs }
{ calculate delta reads }
{ calculate delta writes }
{ compute total reads }
{ compute total writes }
{ for 0 to all the object in the small_class do }
{ compute delta reads per object }
{ compute the sum off all read per object }
{ end }
{ display reads,writes and rates and also read/object class }
{ end }
{ getusermode }
{ end. }
{*****}

#include 'objsmall' $
const
zeros_array=obj_array_type[0,0,0,0,0,0,0,0,0,0];

const
small_object_class_map=1;
max_times=10000;
c_ldev_s=-2; {select all ldev that are disk drives !!!! }

Type
status_type=packed record
```

```

        case integer of
        0:(all:integer);
        1:(
            info:shortint;
            subsys:shortint
        );
        end;

mi_io_ldev_counter_info=packed record
        ldev:integer;
        read:integer;
        write:integer;
        read_obj:obj_array_type;
        end;

Display=string[255];
Var
    loop:integer;
    parm:shortint;
    info:string[255];
    userid:integer;
aif_status:status_type;
item_nums:array[0..4] of integer;
item_statuses:array[0..4] of integer;
mapping:integer;
mi_status:status_type;
item_data:array[0..64] of mi_io_ldev_counter_info;
old_item_data:array[0..64] of mi_io_ldev_counter_info;
buffer_ptr:globalanyptr;
buffer_ptr2:globalanyptr;
wait_time:real;
j1,i,index,loop_count:integer;
old_time:integer;
current_time:integer;
ldev_s:integer;
num_of_ldevs:integer;
delta_read,delta_Write,total_read,total_Write:integer;
delta_obj_read,total_obj_read:obj_array_type;
rinterval:real;
interval:integer;
names:mi_obj_names_type;

$sysintr 'aifintr.pub.sys'$
procedure aifaccesson;intrinsic;
procedure aifaccessoff;intrinsic;

$sysintr 'miintr'$
procedure miioon;intrinsic;
procedure miioget;intrinsic;
procedure miiioff;intrinsic;

$sysintr$

```

```

procedure getprivmode;intrinsic;
procedure getusermode;intrinsic;
procedure pause;intrinsic;
function timer:integer;intrinsic;

$PAGE$
$TITLE 'Write_status - writes status & subsystem error messages'$
Procedure Write_Status(Message:Display;
                      Status:status_type);
BEGIN
WriteLn(Message,' Subsystem: ',Status.subsys:1,
        ' Info: ',Status.Info:1);
END;

begin
userid:=666;
mapping:=small_object_class_map;
{ initialize item_nums array for call }
item_nums[0]:=23009;  {swaplist_read}
item_nums[1]:=23010;  {swaplist_Write}
item_nums[2]:=23050;  {object_class_disk_read }
item_nums[3]:=0;      {terminator}

ldev_s:=c_ldev_s;
buffer_ptr:=addr(item_data);
buffer_ptr2:=addr(old_item_data);
wait_time:=parm;

getprivmode;
aifaccesson(aif_status,userid);
if(aif_status.all<>0 )then
  begin
    write_status(' can''t get the aif          ',aif_status);
    halt(50);
  end;
{ let's turn on the MI for the I/O Disks }

miioon( aif_status,item_nums,item_statuses,mapping,ldev_s);
if(aif_status.all<>0 )then
  begin
    write_status(' can''t get the aif          ',aif_status);
    halt(50);
  end;
old_time:=timer;

item_nums[0]:=20020;  {get ldev }
item_nums[1]:=23009;  {swaplist_read }
item_nums[2]:=23010;  {swaplist_write }
item_nums[3]:=23050;  {obj class counter }
item_nums[4]:=0;

```

```

miiaget(aif_status,item_nums,buffer_ptr2,item_statuses,sizeof(old_item_data),
        mapping,ldev_s,num_of_ldevs);

for loop:=0 to max_times do
    begin
        pause(wait_time);    {pause one second.....}
        j1:=0;i:=0;
        delta_read:=0;
        delta_write:=0;
        total_read:=0;
        total_write:=0;
        delta_obj_read:=zeros_array;
        total_obj_read:=zeros_array;
        current_time:=timer;
        miiaget(aif_status,item_nums,buffer_ptr,item_statuses,sizeof(item_data),
                mapping,ldev_s,num_of_ldevs);

for i:=0 to num_of_ldevs do
    begin
        delta_read:=item_data[i].read-old_item_data[i].read;
        delta_write:=item_data[i].write - old_item_data[i].write;
        for j1:=0 to num_standard_objcl-1 do
            begin
                delta_obj_read[j1]:=item_data[i].read_obj[j1]-old_item_data[i].read_obj[j1];
                total_obj_read[j1]:=total_obj_read[j1]+delta_obj_read[j1];
            end;
        old_item_data[i]:=item_data[i];
        total_read:=total_read + delta_read;
        total_Write:=total_Write + delta_write;
        end;    {end of all ldevs !!!}
        interval:=current_time - old_time;
        rinterval:=interval/1000;
        old_time:=current_time;
        writeln(#27,'h',#27,'J');    {home and clear display }
        writeln ('disk ',
                'read: ',total_read:3,
                ' write: ',total_Write:3,
                ' read rate ',total_read/rinterval:1:1,
                ' write rate ',total_write/rinterval:1:1
                );
        for i:=ord(objcl_transient_data) to ord(objcl_cm_sys_lib) do
            writeln(mi_obj_array[i],total_obj_read[i]);
            writeln(#27,'h');
        end;
    end.

```


Processor Counters

This appendix provides descriptions of the item numbers for the counters that measure processor specific events as well as descriptions of the events measured. These item numbers are valid for `MIPRCSRGET` call.

Table I-1. Processor Counter Information

Num	Counter Name (Data Type) and Description
27000	gmc_idle (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the dispatcher enters the idle state. This state is reached when there are no ready processes and there is no process blocked on disk io.
27001	gmc_mem_mgr_replenish_critical_pool_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the dispatcher, upon receiving a notification message from the memory manager, initiates the system or I/O fetch. This occurs when the memory manager cannot afford to wait until the dispatcher looks at the process and initiates the I/O for that process.
27002	gmc_mem_mgr_sys_fetch_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the dispatcher, upon receiving a notification message from the memory manager, initiates the system or I/O fetch. This occurs when the memory manager cannot afford to wait until the dispatcher looks at the process and initiates the I/O for that process.
27003	gmc_mem_mgr_swapin_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The number of times the dispatcher enters the swapin state. This state is reached when a process blocks for disk I/O and the disk I/O is initiated for the process by the dispatcher.
27004	gmc_mem_mgr_tos_trap_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the dispatcher, upon receiving a notification message from the memory manager, initiates the grey page pool management routines to clean up or to replenish the grey page pool for tos trap handling.
27005	gmc_dispatcher_ics (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the dispatcher is active (executing its own code), not on behalf of the memory manager or the user process, such as scanning the dispatcher queue for a launchable process or for swap-ins.
27006	gmt_idle (I32) Counter Type: Simple Counter Unit of Measurement: ML Time The total length of time that the dispatcher was in the idle state. This state is reached when there are no ready processes and there is no process blocked on disk I/O.

I-2 Processor Counters

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27007	gmt_mem_mgr_replenish_critical_pool_ics (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total number of times that the dispatcher, upon receiving a notification message from the memory manager, initiates the system or I/O fetch. This occurs when the memory manager cannot afford to wait until the dispatcher looks at the process and initiates the I/O for that process.
27008	gmt_mem_mgr_sys_fetch_ics (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total CPU time charged to the memory manager to fetch in a page. This occurs when the dispatcher, upon receiving a notification message from the memory manager, initiates the system or I/O fetch.
27009	gmt_mem_mgr_swapin_ics (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The length of time the dispatcher was in the swapin state. This state is reached when a process blocks for disk I/O and the disk I/O is initiated for the process by the dispatcher.
27010	gmt_mem_mgr_tos_trap_ics (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of CPU time charged to the memory manager to handle TOS trap. This occurs when the dispatcher upon receiving a notification message from the memory manager, initiates the grey page pool management routines to clean up or to replenish the grey page pool for tos trap handling.
27011	gmt_dispatcher_ics (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of CPU time charged to the dispatcher to handle its own management routines. This occurs when the dispatcher is active (executing its own code), not on behalf of the memory manager or the user process, such as scanning the dispatcher queue for a launchable process or for swap_ins.
27012	gmt_external_ics (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total length of CPU time that the interrupt handler first executes on the ICS before it transfers control to the dispatcher.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27013	gmc_launches (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the dispatcher chooses a process from the ready queue and launches the process (gives control of the CPU to the process).
27014	gmc_interrupt_handler (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times the system switches to the ICS.
27015	gmc_cpu_completion (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of processes that have terminated.
27016	gmc_as_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes block while running in the AS queue.
27017	gmc_bs_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes block while running in the BS queue.
27018	gmc_cs_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes block while running in the CS queue.
27019	gmc_ds_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes block while running in the DS queue.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27020	gmc_es_process (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes block while running in the ES queue.
27021	gmt_as_process (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total CPU time used by processes running in the AS queue.
27022	gmt_bs_process (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total CPU time used by processes running in the BS queue.
27023	gmt_cs_process (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total CPU time used by processes running in the CS queue.
27024	gmt_ds_process (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total CPU time used by processes running in the DS queue.
27025	gmt_es_process (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total CPU time used by processes running in the ES queue.
27026	gmt_cm_time (I32) Counter Type: Simple Counter Unit of Measurement: MI_Time The total time spent by the system in compatibility mode (CM).
27027	gmc_stop_nm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a native mode code page fault.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27028	gmc_stop_nm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a native mode stack page fault.
27029	gmc_stop_nm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a native mode transient page fault (heap, swappable table).
27030	gmc_stop_file_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a file page fault.
27031	gmc_stop_cm_code_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a compatible mode code page fault.
27032	gmc_stop_cm_stack_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a compatible mode stack page fault.
27033	gmc_stop_cm_trans_page_fault (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for an I/O to complete on a compatibility mode transient page fault.
27034	gmc_stop_term_read (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a terminal read.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27035	gmc_stop_term_write (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes wait for a terminal write such as console I/O or gen message for WARN and TELL messages. They generate direct ATTACHIO calls and bypass the file system.
27036	gmc_stop_disc_io (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the process blocked for disk I/O.
27037	gmc_stop_other_io (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for non-disc I/O devices.
27038	gmc_stop_ipc_trans_complete (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked on a port with transaction completed as one of their options(for example, the IOWAIT intrinsic).
27039	gmc_stop_sir_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a SIR.
27040	gmc_stop_rin_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a RIN.
27041	gmc_stop_mem_mgr_prefetch (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a memory manager prefetch.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27042	gmc_stop_quantum_expiration (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes use up their time quantum.
27043	gmc_stop_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a timeout or pause with one second or less.
27044	gmc_stop_parent_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes are waiting for their parents to wake them up.
27045	gmc_stop_cntl_block_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes are blocked for a control block on a semaphore.
27046	gmc_stop_child_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes are blocked for their children to wake them up.
27047	gmc_stop_data_comm_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes are blocked for data communication.
27048	gmc_stop_rit_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a RIT (operator reply).

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27049	<p>gmc_stop_disp_work (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes were preempted by the dispatcher to work on higher-priority system processes (power failure, grey page cleanup, replenish critical pool, fetch I/O, or system fetch).</p>
27050	<p>gmc_stop_port_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a port (default IPC wait).</p>
27051	<p>gmc_stop_mail_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a port for a MAIL (old type of IPC interface that existed prior to message file implementation).</p>
27052	<p>gmc_stop_junk_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a port for a JUNK wait (don't care type wait).</p>
27053	<p>gmc_stop_message_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a port for a MESSAGE (basic IPC message file).</p>
27054	<p>gmc_stop_impede (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes impeded (mostly used by the file system for synchronization purposes).</p>
27055	<p>gmc_stop_break_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes are in break mode.</p>

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27056	gmc_stop_wait_queue (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked to be put on a wait queue by PORTS (table management running out of entries waits for additional spaces to be allocated).
27057	gmc_stop_memmgt_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the memory manager waits for proper I/O synchronization, excluding user I/O request such as POST.
27058	gmc_stop_port_blocked_make_present (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked until the requested port is present.
27059	gmc_stop_file_blocked (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked on a port for posting pages through the call CM_POST.
27060	gmc_stop_file_unblocked (I32) Counter Type: Simple Counter Unit of Measurement: Count This block on port is not used. This counter should be zero.
27061	gmc_stop_storage_mgt (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times processes blocked on a port through storage management.
27062	gmc_stop_user_to_debug_msg (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes (CIs) blocked due to breakpoint contention.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27063	gmc_stop_io_config_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked because devices are being configured or released.
27064	gmc_stop_pfp_reply_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked because the port facility process needs to be created, initialized, or checked.
27065	gmc_stop_db_monitor_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that the database monitor (SQL) waits for the DB CLEAN_UP process to finish cleaning up the aborted DB processes before closing the database.
27066	gmc_stop_fill_disc_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that HLIO waits for the master MIB because the new file extent in secondary storage needs to be initialized with fill characters for all virgin pages.
27067	gmc_stop_hlio_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes are blocked because HLIO aborts the I/O.
27068	gmc_stop_fs_tio_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that TIO (Terminal I/O) Fast Write in DTS (BND LDM) waits for a reply message from the device manager or the buffer management.
27069	gmc_stop_mem_mgr_post_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for I/O completion when explicitly posting pages to disk.

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27070	gmc_stop_signal_timer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked for a delay or timer on a standard signal port.
27071	gmc_stop_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes get preempted by a higher-priority process due to process awakening (IPC wait other than disk I/O completion).
27072	gmc_stop_disc_io_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes get preempted by a higher-priority process due to disk I/O completion. This includes page fault, post, and prefetch.
27073	gmc_stop_priority_preemption (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes get preempted by a higher-priority process due to priority boosting or dropping.
27074	gmc_stop_sql_lock_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of processes that are blocked to acquire a SQL lock. This lock is required for user data (a tuple, a page, or a relation).
27075	gmc_stop_sql_latch_level_1_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked on a level 1 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).
27076	gmc_stop_sql_latch_level_2_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that processes blocked on a level 2 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27077	<p>gmc_stop_sql_latch_level_3_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 3 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27078	<p>gmc_stop_sql_latch_level_4_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 4 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27079	<p>gmc_stop_sql_buffer_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number times that processes blocked to change the state of a page in SQL buffer. The buffer is used to hold a page (4 K) of user data. The page can be in a number of states (for example, being updated, in transit in, in transit out of). When a process requests a page be placed in a state that conflicts with its current state, the process blocks.</p>
27080	<p>gmc_stop_long_pause_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes paused for 2 secs or more.</p>
27081	<p>gmc_stop_mem_mgr_freeze_and_other (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes block on freeze and corner cases other than page_fault, prefetch, and freeze. So this counter is predominantly blocked on freeze.</p>
27082	<p>gmc_stop_other_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes wait for other block reasons such as giving up CPU because processes weren't removed accordingly. This includes all reasons other than that are measured.</p>

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27083	<p>gmc_stop_sql_latch_level_5_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 5 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27084	<p>gmc_stop_sql_latch_level_6_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 6 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27085	<p>gmc_stop_sql_latch_level_7_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 7 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27086	<p>gmc_stop_sql_latch_level_8_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 8 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27087	<p>gmc_stop_sql_latch_level_9_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 9 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27088	<p>gmc_stop_sql_latch_level_10_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 10 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27089	<p>gmc_stop_sql_latch_level_11_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 11 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27090	<p>gmc_stop_sql_latch_level_12_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 12 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27091	<p>gmc_stop_sql_latch_level_13_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 13 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27092	<p>gmc_stop_sql_latch_level_14_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 14 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27093	<p>gmc_stop_sql_latch_level_15_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 15 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27094	<p>gmc_stop_sql_latch_level_16_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 16 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27095	<p>gmc_stop_sql_latch_level_17_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 17 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27096	<p>gmc_stop_sql_latch_level_18_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 18 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27097	<p>gmc_stop_sql_latch_level_19_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 19 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27098	<p>gmc_stop_sql_latch_level_20_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 20 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27099	<p>gmc_stop_sql_latch_level_21_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 21 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27100	<p>gmc_stop_sql_latch_level_22_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 22 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27101	<p>gmc_stop_sql_latch_level_23_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 23 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27102	<p>gmc_stop_sql_latch_level_24_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 24 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27103	<p>gmc_stop_sql_latch_level_25_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 25 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27104	<p>gmc_stop_sql_latch_level_26_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 26 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27105	<p>gmc_stop_sql_latch_level_27_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 27 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27106	<p>gmc_stop_sql_latch_level_28_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 28 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27107	<p>gmc_stop_sql_latch_level_29_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 29 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27108	<p>gmc_stop_sql_latch_level_30_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 30 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27109	<p>gmc_stop_sql_latch_level_31_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 31 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>
27110	<p>gmc_stop_sql_latch_level_32_wait (I32) Counter Type: Simple Counter Unit of Measurement: Count</p> <p>The total number of times that processes blocked on a level 32 latch. A latch is used to coordinate access to its run-time data structures. Each latch has a level associated with it (used for deadlock prevention).</p>

Table I-1. Processor Counter Information (continued)

Num	Counter Name (Data Type) and Description
27111	gmc_stop_release (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that a process has given up the CPU during process termination. The usual value of the counter is zero.
27112	gmc_stop_memmgt_psuedo_ioread (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that a process has stopped with a block_and_send event descriptor and is initiating a read.
27113	gmc_stop_memmgt_psuedo_iowrite (I32) Counter Type: Simple Counter Unit of Measurement: Count The total number of times that a process has stopped with a block_and_send event descriptor and is initiating a write.

Index

- A**
 - access management, 3-1
 - acquiring global counter information, 5-20
 - acquiring I/O counter information, 5-25
 - acquiring process counter information, 5-36
 - acquiring processor counter information, 5-33
 - acquiring system configuration information, 3-2
 - activate user ID, 5-3
 - address type, 4-1
 - AIFACCESSOFF, 3-1, 5-2
 - AIFACCESSION, 3-1, 5-3
 - AIFGLOBINSTALL, 1-6, 3-5, 5-4
 - AIFKUF file, 1-3
 - AIFSCGET, 5-5
 - AIFSCPUT, 5-7
 - AIFTIME, 3-5, 5-18
 - ALLOW command, 3-3
 - ALLOW mask, 3-3
 - applications
 - shipping with AIFs, 1-5
 - Architected Interface Facility, 1-1
 - Measurement Interface:installation, 1-3
 - Measurement Interface:product tape , 1-4
 - architected interfaces, 1-1
 - access management, 3-1
 - acquiring global counter information, 5-20
 - acquiring I/O counter information, 5-25
 - acquiring process counter information, 5-36
 - acquiring processor counter information, 5-33
 - activating user ID, 5-3
 - AIFACCESSOFF, 3-1, 5-2
 - AIFACCESSION, 3-1, 5-3
 - AIFSCGET, 3-3, 5-5
 - AIFSCPUT, 3-3, 5-7
 - AIFTIME, 5-18
 - calling AIFs efficiently, 3-1
 - customer, 1-2
 - data structures, B-1
 - data types, 4-1
 - declaration files, 1-4
 - declaring, 4-1
 - defined, 1-1
 - design strategy, 1-2
 - disabling global counter information, 5-22
 - disabling I/O counter information, 5-28
 - disabling process counter information, 5-40

- enabling global counter information, 5-23
- enabling I/O counter information, 5-30
- enabling process counter information, 5-41
- error management, 4-3
- error messages, A-1
- examples, H-1
- GET , 4-2
- getting configuration information, 5-5
- getting time information, 5-18
- global class counters, E-1
- global information, 3-3
- hardware requirements, 1-1
- information access, 3-1, 3-2
- installing, 1-3
- intended use, 1-2
- introduction, 1-1
- I/O counter descriptions, G-1
- item status, 4-3
- item verification, 4-4
- MIGLOBGET, 3-3, 5-20
- MIGLOBOFF, 3-3, 5-22
- MIGLOBON, 3-3, 5-23
- MIIOGET, 3-4, 5-25
- MIIIOFF, 3-4, 5-28
- MIIION, 3-4, 5-30
- MIPRCSRGET, 3-4, 5-33
- MIPROCGET, 3-4, 5-36
- MIPROCOFF, 3-4, 5-40
- MIPROCON, 3-4, 5-41
- modifying configuration information, 5-7
- privileged mode, 1-2
- process counter descriptions, F-1
- process information, 3-4
- processor class counters, I-1
- PUT, 4-2
- semaphore counter descriptions, D-1
- shipping products, 1-5
- software requirements, 1-1
- status messages, A-1
- subsystem number, 4-3
- tracking product changes, 2-5
- use, 4-2
- arrays
 - dynamic length, 4-2
- array type, 4-1, 4-2
- A type, 4-1

- B** B type, 4-1
- C** changes to architected interfaces, 2-5
 - C language, 1-1, 4-2
 - declaring MI AIFs, 1-3
 - clock_type, B-1
 - compilers supported, 1-1
 - configuration information, 3-2, 3-3
 - acquiring, 5-5
 - modifying, 5-7
 - counter rollover, 2-4
 - cycle time, 2-5
 - detecting, 2-4
 - minint and maxint, 2-5
 - recovering from, 2-4
 - counters, 2-1
 - acquiring global counter information, 5-20
 - acquiring information about, 3-3
 - acquiring I/O counter information, 5-25
 - acquiring process counter information, 5-36
 - acquiring processor counter information, 5-33
 - disabling, 3-3
 - disabling global counter information, 5-22
 - disabling I/O counter information, 5-28
 - disabling process counter information, 5-40
 - enabling, 3-3
 - enabling global counter information, 5-23
 - enabling I/O counter information, 5-30
 - enabling process counter information, 5-41
 - frequency, 2-1
 - global class descriptions, E-1
 - holding larges value to date, 2-1
 - I/O counter descriptions, G-1
 - mapping, 2-3
 - measuring, 2-2, 2-4
 - object class descriptions, C-1
 - object classes, 2-1, 2-2
 - organization, 2-1
 - process descriptions, F-1
 - processor class descriptions, I-1
 - rollover, 2-4
 - semaphore class, 2-1
 - semaphore class descriptions, D-1
 - time stamp, 2-4
 - unit of measurement, 2-4
 - C type, 4-1
 - customers defined, 1-1, 1-2

- D**
 - data structures, B-1
 - data type mapping, 4-1
 - data types, 4-1, B-1
 - mapping to languages, 4-2
 - datestr_type, B-1
 - date_type, B-2
 - deactivate user ID, 5-2
 - declaring architected interfaces, 4-1
 - declaring MI AIFs, 1-3
 - detecting counter rollover, 2-4
 - device counters, 3-4
 - device_name_type, B-2
 - directory_name_type, B-2
 - disabling counters, 3-3
 - disabling device counters, 3-4
 - disabling global counters, 5-22
 - disabling I/O counter information, 5-28
 - disabling process counter information, 5-40
 - disabling processor counters, 3-4
 - documentation files, 1-4
 - DOBJCL, 1-4
 - DSEMCL, 1-4
 - GLEIN, 1-4
 - IOEIN, 1-4
 - PREIN, 1-4
 - DSEMCL file, 1-4
 - dstsrec_type, B-3
 - dynamic length arrays, 4-2

- E**
 - enabling counters, 3-3
 - enabling device counters, 3-4
 - enabling global counters, 5-23
 - enabling I/O counter information, 5-30
 - enabling process counter information, 5-41
 - enabling processor counters, 3-4
 - error checking, 4-3
 - item status, 4-3
 - overall status, 4-3
 - performance, 4-3
 - subsystem number, 4-3
 - verification item status, 4-4
 - error message summary, A-1
 - examples, H-1

- F**
 - filename_type, B-3
 - files
 - declarations, 1-4
 - DOBJCL, 1-4
 - DSEMCL, 1-4
 - GLEIN, 1-4
 - intrinsic definitions, 1-3
 - IOEIN, 1-4
 - MIINTR, 1-3
 - PREIN, 1-4

fnumpid_type, B-3

- G**
 - generic data types, 4-1
 - GET AIFs, 3-3
 - getting I/O counter information, 5-25
 - getting system configuration information, 3-2
 - global counter descriptions, E-1
 - global counter information, 3-2, 3-3
 - acquiring, 5-20
 - disabling, 5-22
 - enabling, 5-23
 - global events, 2-2
 - granularity of measurement, 2-4

- H**
 - hardware requirements, 1-1
 - histogram set, 2-1

- I**
 - I32 type, 4-1
 - information access, 3-1, 3-2
 - installation
 - AIFGLOBINSTALL, 1-6, 5-4
 - AIFKUF file, 1-3
 - INSTMI utility, 1-3
 - MIINTR file, 1-3
 - user ID, 1-3, 3-2
 - installing measurement interface AIFs, 1-3
 - installing products
 - AIFGLOBINSTALL, 5-4
 - INSTMI
 - programmatic interface, 5-4
 - INSTMI utility, 1-3, 3-2
 - intrinsic definitions, 1-3
 - intrinsic definitions, 1-3
 - intrinsic definitions, 1-3
 - intrinsic definitions, 1-3
 - I/O counter descriptions, G-1
 - I/O counter information, 3-2
 - acquiring, 5-25
 - disabling, 5-28
 - enabling, 5-30
 - I/O events, 2-2
 - I/O information, 3-4
 - item_array_type, B-4
 - item descriptions
 - system configuration, 5-10
 - itemnum_array_type, B-4
 - item status, 4-3
 - item_status_array_type, B-4
 - item verification, 4-4

J jsdev_type, B-5
 jskey_type, B-5
 jsnum_type, B-5

L large map
 defined, 2-3
 logon_desc_type, B-6
 longint_type, B-6
 long pointer address, 4-1

M machine ticks, 2-4
 magnetic tape, 1-4
 mapping, 2-3
 defined, 2-3
 enabling, 2-3
 large, 2-3, C-1
 small, 2-3, C-1
 standard, 2-3, C-1
 maxint, 2-5
 measurement
 count, 2-4
 counter rollover, 2-4
 granularity, 2-4
 rounding factor, 2-4
 ticks, 2-4
 time, 2-4
 measurement interface
 AIF types, 3-1
 counters, 2-1
 intrinsic definitions, 1-3
 object class counters, 2-1
 semaphore class, 2-3
 semaphore class counters, 2-1
 measurement interface AIFs
 installing, 1-3
 message_buffer_type, B-6
 MI AIF document files, 1-4
 MIGLOBGET, 3-3, 5-20
 MIGLOBOFF, 3-3, 5-22
 MIGLOBON, 3-3, 5-23
 MIINTR file, 1-3
 MIIOGET, 3-4, 5-25
 MIIOFF, 3-4, 5-28
 MIIOON, 3-4, 5-30
 minint, 2-5
 MIPCSRGET, 3-4, 5-33
 MIPROCGET, 3-4, 5-36
 MIPROCOFF, 3-4, 5-40
 MIPROCON, 3-4, 5-41

- N** naming conventions, 4-1

- O**
 - object class
 - defined, 2-2
 - object class counters, 2-1, 2-3
 - descriptions, C-1
 - summary, C-1
 - OFF AIFs, 3-3
 - ON AIFs, 3-3
 - organization of counters, 2-1
 - overall status, 4-3

- P**
 - pac16, B-7
 - pac18, B-7
 - pac256, B-8
 - pac32, B-7
 - pac34, B-8
 - pac8, B-7
 - P and V primitives, 2-3
 - parameter data types, 4-1
 - parameters
 - overall status, 4-3
 - Pascal language, 1-1, 4-2
 - declaring MI AIFs, 1-3
 - performance, 3-1, 4-3
 - performance concerns, 1-2
 - pid_type, B-8
 - privileged mode, 1-1, 1-2, 4-3
 - procedures for installation, 1-3
 - process counter descriptions, F-1
 - process counter information, 3-2
 - acquiring, 5-36
 - disabling, 5-40
 - enabling, 5-41
 - process events, 2-2
 - process information, 3-4
 - Process Information, 3-4
 - processor counter descriptions, I-1
 - processor counter information, 3-2
 - acquiring, 5-33
 - processor counters, 3-4
 - processor events, 2-2
 - Processor information, 3-4
 - product installation, 5-4

- R**
 - recfnumpid_type, B-8
 - record type, 4-1, 4-2
 - recovering from counter rollover, 2-4
 - risks, 1-2
 - rounding factor, 2-4

- S**
 - search_key_type, B-9
 - sel_eq_type, B-9
 - semaphore class, 2-3
 - defined, 2-3
 - semaphore class counter descriptions, D-1
 - semaphore class counters, 2-1
 - shipping products, 1-5
 - short pointer address, 4-1
 - small map
 - defined, 2-3
 - software requirements, 1-1
 - spf_id_type, B-10
 - standard map
 - defined, 2-3
 - status message summary, A-1
 - status_type, B-10
 - status_type record, 4-3
 - subsystem number, 4-3
 - supporting AIFs, 3-2
 - synchronization, 2-3
 - system configuration information, 3-2, 3-3
 - acquiring, 5-5
 - modifying, 5-7
 - system configuration item descriptions, 5-10
 - system variables, 3-3
 - system-wide configuration information, 3-3

- T**
 - tick factor, 2-4
 - ticks, 2-4
 - time information
 - acquiring, 5-18
 - time stamp, 2-4
 - tracking product changes, 2-5
 - TUNE command, 3-3

- U**
 - U32 type, 4-1
 - ufid_type, B-10
 - user, 1-2
 - user ID, 3-1, 3-2
 - user ID installation, 1-3
 - user IDs
 - activating, 5-3
 - deactivating, 5-2
 - using with AIFs, 3-1
 - utilities
 - AIFGLOBINSTALL, 3-5
 - AIFTIME, 3-5

- V** validating user access, 3-1
- variables
 - system, 3-3
- vendor ID, 1-3
- verification item status, 4-4
- version number, 2-5
- virtual space, 2-2

