900 Series HP 3000 Computer Systems

MPE/iX Architected Interface Facility: Operating System Reference Manual

HP Part No. 36374-90001
Printed in U.S.A. 1995

Sixth Edition
E0195

Architected Interfaces Notice

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	April 1990	A.00.01
Second Edition	December 1990	A.01.01
Third Edition	June 1992	B.40.00
Fourth Edition	October 1992	C.45.00
Fifth Edition	April 1994	C.50.00
Sixth Edition	January 1995	C.50.00

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: Operating System Reference Manual (36374-90001) is written for the experienced programmer.

This manual is organized into the following chapters and appendixes:

Chapter 1 **Introduction** contains an introductory overview of architected interfaces in general and operating system architected interfaces (AIFs) in particular, as well as installation procedures.

Chapter 2 **Using Operating System Architected Interfaces** lists data type naming conventions and the Architected Interface Facility error management strategy.

Chapter 3 **Architected Interface Descriptions** lists the architected interfaces and their syntax.

Appendix A **AIF Status Messages** contains a list of all the error messages returned by operating system architected interfaces.

Appendix B **AIF Data Structures** contains a list of all the data structures used in the architected interfaces.

Appendix C **Programming Examples** contains AIF programming examples written in Pascal and C.

Appendix D **Glossary** provides definitions of terms used in this manual.

The following manual contains more information on Operating System Architected Interfaces:

- *MPE XL Architected Interface Facility: Operating System Self-Paced Training Workbook* (36374-90002)

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:

```
COMMAND
```

can be entered as any of the following:

```
command        Command        COMMAND
```

It cannot, however, be entered as:

```
comm           com_mand       comamnd
```

italics 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 *filename* with the name of the file:

`COMMAND` *filename*

bold italics 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 ***filename*** with the name of the file:

`COMMAND(`***filename***`)`

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:

(*filename*):(*filename*)

underlining Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, yes is the user's response to the prompt:

```
Do you want to continue? >>  yes
```

`{ }` 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`:

`COMMAND` $\left\{ \begin{array}{l} \texttt{ON} \\ \texttt{OFF} \end{array} \right\}$

`[ ]` In a syntax statement, brackets enclose optional elements. In the following example, `OPTION` can be omitted:

`COMMAND` *filename* `[OPTION]`

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 *parameter* or neither. The elements cannot be repeated.

`COMMAND` *filename* $\left[\begin{array}{l} \texttt{OPTION} \\ \textit{parameter} \end{array} \right]$

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.

$$\left\{ \begin{matrix} \texttt{A} \\ \texttt{B} \end{matrix} \right\} \texttt{| ... |}$$

`...`

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, RETURN represents the carriage return key or Shift represents the shift key.

CTRL*character*

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

Contents

Tables

1

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 abstract the operating system structures. 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 most 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.

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.

Note

Architected Interface Facility products provide supported AIFs without the commitment to backward compatibility as with system intrinsics.

For example, many MPE/iX intrinsics were included just to ensure backward compatibility with MPE V. New MPE/iX intrinsics provide the same backward compatibility over the life of MPE/iX. On the other hand, AIFs may change over the life of MPE/iX to reflect changes to system internals. AIF design will be consistent over time, but data returned or the functions provided will change as underlying operating system data structures and functionality change.

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.

Installing Operating System Architected Interfaces

The Architected Interface Facility: Operating System product includes the following files:

- INSTOS
- AIFINTR

INSTOS

INSTOS is an installation utility that enables you to execute operating system AIF code located on the 900 Series HP 3000 computer system, using the user ID that INSTOS has installed. INSTOS must be executed on all systems containing code that calls operating system AIFs (for example, your application). INSTOS 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: Operating System product.

Note

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

AIFINTR

The AIFINTR file is a binary file that contains the intrinsic definitions of all operating system AIFs. Use AIFINTR in your program to declare operating system AIFs.

Following is an example of Pascal code that enables the HP Pascal/iX compiler to locate the operating system AIF intrinsic file AIFINTR:

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

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

Following is an example of C code that enables the HP C/iX compiler to locate the operating system AIF intrinsic file AIFINTR:

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

How to Ship Products That Use Operating System Architected Interfaces

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

- Use the `INSTOS` 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 `INSTOS` utility.

Using INSTOS

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

1. Develop the code according to the guidelines specified above.
2. Make the file `INSTOS` a part of the final product by shipping it along with your code.
3. Include these steps in the installation procedures for your application:
   a. Restore the file `INSTOS` into the target system's `PUB.SYS`.
   b. Execute the program `INSTOS` to install your user ID onto the target system.
   c. Purge the file `INSTOS` 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

The `AIFGLOBINSTALL` AIF is the programmatic equivalent of executing the `INSTOS` installation utility. `AIFGLOBINSTALL` must be executed on all systems containing code that calls operating system AIFs (for example, your application). It should be executed once per installation; however, it can be executed each time that your application is run without side effects. Your application must execute `AIFGLOBINSTALL` prior to calling any other operating system AIFs.

The `AIFGLOBINSTALL` AIF fails 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 reexecute code that contains the call to `AIFGLOBINSTALL`.

Types of Operating System Architected Interfaces

The MPE/iX Architected Interface Facility: Operating System product provides three types of AIFs:

- Access management AIFs
- Information access AIFs
- Functional access AIFs

Access Management Architected Interfaces

Access management AIFs provide a mechanism, the user ID, to validate user access to operating system AIFs.

User IDs

Each purchaser of the Architected Interface Facility: Operating System product is assigned a unique user ID. Whenever you call an AIF, you should identify yourself by using your company's user ID.

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 are assumed to belong to the same user ID. Use `AIFACCESSOFF` after completing the multiple AIF calls.

Note

You must use the user ID installed through `INSTOS` in all calls to AIFs described in this manual. If you have purchased another Architected Interface Facility product (for example, measurement interface AIFs), you still need to call `AIFACCESSON` with the Architected Interface Facility: Operating System user ID in order to call operating system AIFs, even if you have called `AIFACCESSON` 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 MPE/iX internal table information while abstracting the structure from the user.

The information access AIFs provide a single AIF, `AIFSYSWIDEGET`, that is normally the starting point for information retrieval. `AIFSYSWIDEGET` returns information on the current state of the system. For example, it can provide a list of objects that currently exist on the system and meet a specified set of criteria. The information provided by `AIFSYSWIDEGET` can be passed to the other information access AIFs so that more detailed information can be acquired.

Information access AIFs can be used without first using `AIFSYSWIDEGET`. For example, you can call a global file information AIF by passing a known file name.

Each information access AIF attempts to lock all of the tables associated with that object.

Information Get and Put

In most cases, there are two types of AIF for each class of objects that information can be accessed, an `AIF`*nn*`GET` and an `AIF`*nn*`PUT`.

The `AIF`*nn*`GET` AIF returns information about a specific object identified by the input keys. All `AIF`*nn*`GET` 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 *itemstatus_array* parameter for the items in error, while the rest of the item values are returned normally.

The `AIF`*nn*`PUT` AIFs update relevant tables to obtain a consistent, updated state of the system. Only a subset of the items provided by the `AIF`*nn*`GET` AIFs are available to `AIF`*nn*`PUT` AIFs.

Information Verification

Because there is no guarantee that the information that is returned from an `AIF`*nn*`GET` AIF is still valid when an `AIF`*nn*`PUT` AIF is called, each `AIF`*nn*`PUT` AIF allows for verification of values to take place before a system table update occurs.

A list of items and corresponding values may be specified in the verification arrays to the `AIF`*nn*`PUT` AIF. Each item is checked and validated before attempting to do a system table update. If any single item that is to be verified fails, the information is not placed into the system. This verification helps prevent the system from accidentally being placed into an undesirable state. If no verification items are specified, the system table update is performed unconditionally.

System-Wide Information

The system wide information AIF is

- AIFSYSWIDEGET

The AIFSYSWIDEGET AIF is normally the first AIF called. It returns information about a whole class of objects, instead of information about a particular object as the other AIFs do.

The AIFSYSWIDEGET AIF enables you to specify an object class as well as a list of criteria that you wish to apply to the objects in that class. It applies all of the criteria to each object located, returning only those objects that meet the criteria that you specify. The AIFSYSWIDEGET AIF returns a list of meaningful object identifiers and, optionally, a corresponding list of alternative input keys, when possible. You can use the alternative input keys with other AIFs to retrieve information faster than using the object identifiers.

The AIFSYSWIDEGET AIF may return a context key that indicates that there are a greater number of objects available than there are elements in the user-defined array passed to the AIF. Use this context key in a subsequent call to retrieve the additional objects.

Accounting Information

The accounting information AIFs are

- AIFACCTGET
- AIFACCTPUT

Accounting information AIFs return or update information associated with directory objects such as users, groups, and accounts. Accounting information AIFs use a user name, group name, or account name as input keys. The input keys default to the calling process' user name, group name, and account name.

File Information

There are two types of file information AIFs:

- local file information AIFs
- global file information AIFs

Local File Information. Local file information AIFs are

- AIFFILELGET
- AIFFILELPUT

Local file information AIFs use a PID and a process-specific file number as input keys, with a UFID for validation. The AIFFILELGET AIF returns PIDs of the sharers of the file and the file numbers. The returned information can be used with subsequent calls to process information AIFs, global file information AIFs, or other local file information AIFs.

Global File Information. Global file information AIFs are

- AIFFILEGGET
- AIFFILEGPUT

Global file information AIFs use file names and UFIDs as input keys.

Job or Session Information

Job or session information AIFs are

- AIFJSGET
- AIFJSPUT

Job or session AIFs return or update information associated with jobs and sessions. They accept job numbers or session numbers as input keys, returning or updating job or session information associated with the keys. The keys can be obtained either from AIFSYSWIDEGET or from other means (for example, from the SHOWJOB command). The input keys default to the calling process's job or session number.

Process Information

Process information AIFs are

- AIFPROCGET
- AIFPROCPUT

Process information AIFs accept PIDs and PINs as input keys. They return or update process-related information. The input keys can be obtained either from AIFSYSWIDEGET or from file information AIFs.

The AIFPROCGET AIF returns information about all of the files opened by the process, process-specific file numbers, and UFIDs. These three can then be used to query the file interfaces. The AIFPROCGET AIF also returns the default account and group for the process, their names, and UFIDs. This information can be used to query accounting information AIFs.

Reply Information

The reply information AIF requires only a reply request ID as input. It retrieves information on a specified pending reply request. In addition to providing the table information, it also formats the request message as it is displayed on the console by the RECALL command.

The reply information AIF is

- AIFREPLYGET

Spooler Information

There are two types of spooler information AIFs

- spool file information AIFs
- spooler process AIFs

Spool File Information. Spool file information AIFs are

- AIFSPFGET
- AIFSPFPUT

The AIFSPFGET and AIFSPFPUT AIFs accept a file name or an address as input keys and return or update information about files that have been spooled.

Spooler Process Information. Spooler process information AIFs are

- AIFSPPGET
- AIFSPPPUT

The AIFSPPGET and AIFSPPPUT AIFs accept device names as input keys and return or update information about spooler processes.

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 quantums. The dynamic system information includes the next job/session number to be allocated, the CS average quantum for transactions, and the current ALLOW mask.

Device Information

Device information AIFs are:

- AIFDEVCLASSGET
- AIFDEVICEGET
- AIFDEVICEPUT

AIFDEVCLASSGET retrieves information on a specific device class. AIFDEVICEGET retrieves information on a specific device (ldev). AIFDEVICEPUT updates information on a specific device (ldev).

Functionality Access Architected Interfaces

The Architected Interface Facility: Operating System product provides AIFs to manage special functionality normally available only to operating system internals. The types of functionality access provided are:

- User global area management
- Ports management
- Spooler management
- Magneto-Optical Disk Library System
- Miscellaneous utilities

While their external appearance all reflect AIF design standards, each type differs according to the functionality the AIFs provide access to.

User Global Area Management

User global area management AIFs are:

- `AIFGLOBACQ`
- `AIFGLOBGET`
- `AIFGLOBLOCK`
- `AIFGLOBPUT`
- `AIFGLOBREL`
- `AIFGLOBUNLOCK`

User global area management AIFs enable you to share data between multiple processes and enforce concurrence on access to this data.

Ports Management

Ports management AIFs are:

- `AIFPORTCLOSE`
- `AIFPORTOPEN`
- `AIFPORTRECEIVE`
- `AIFPORTSEND`
- `AIFPORTINT`

Ports management AIFs enable you to create and manage Architected Interface Facility user ports. User ports can be used as a fast means of interprocess communication by sending messages from one process to another. User ports do not interfere with or provide access to system ports.

Spooler Management

Spooler management AIFs are:

- AIFSPFLINK
- AIFSPFLIST
- AIFSPPOPENQ
- AIFSPPRELEASE
- AIFSPPRESUME
- AIFSPPSHUTQ
- AIFSPPSTART
- AIFSPPSTOP
- AIFSPPSUSPEND

Spooler management AIFs enable you to manage spool files and spooler processes. For example, you can start, stop, resume, or suspend devices. In addition, you can link files to the MPE/iX spooler facility.

Magneto-Optical Disk Library System

Magneto-Optical Disk Library System AIFs are:

- AIFMOALLOCATE
- AIFMODEALLOCATE
- AIFMODISMOUNT
- AIFMOGET
- AIFMOPUT
- AIFMOMOUNT

Magneto-Optical Disk Library System AIFs provide a supported external interface to optical disk library systems while extracting internal detail. These AIF's provide a layer above the existing Media Manager routines. The Media Manager is a set of operating system routines that are used to control a Magneto-Optical Disk Library System. The Media Manager functions that are provided through AIF's include: allocating and deallocating an optical media drive, mounting and dismounting optical media, and retrieving and modifying optical disk library system information.

Refer to the following manuals for more information:

- *Installing and Using the Optical Disk Library System* (C1700-90076)
- *Magneto-Optical Media Management User's Guide* (36398-90001)

Utilities

Utility AIFs provide miscellaneous functionality useful to application developers.

- `AIFCHANGELOGON` enables you to change the logon environment of a process.
- `AIFCLOSE` enables you to save files across account boundaries.
- `AIFCONVADDR` converts compatibility mode relative addresses to corresponding native mode virtual addresses.
- `AIFGLOBINSTALL` is the programmatic equivalent of executing the `INSTOS` installation utility.
- `AIFTIME` converts ticks and microseconds to a more meaningful time, such as date time, clock time, or a string format.

2

Using Operating System Architected Interfaces

This chapter provides information required for the correct use of operating system architected interfaces. Included in this chapter are discussions about:

- Data type naming conventions used in this manual
- Data type mappings to languages
- Error management

Note

Please read and understand fully the information provided in this chapter before using operating system architected interfaces.

Data Type Naming Convention

Architected interface descriptions specify all parameters and their required data types. Following are listed the generic data types and their mnemonics used in this manual. These data types should be used to declare the types for the parameters and the values passed or returned.

Table 2-1. Data Type Naming Convention

Mnemonic	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 ASCII characters; `BA` represents an array of booleans.
`REC`	Record. Some parameters require complex record structures to be passed. Record structures required by architected interfaces are described in appendix B.

Data Type Mappings to Languages

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

For record types, appendix B provides the Pascal record declarations as well as the packing of the fields as implemented by the HP Pascal/iX compiler. This information should suffice to make the call usable from both Pascal and C.

For array types, AIFs allow you 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, this record structure is merely an extension of the way strings are implemented on most Pascal compilers today. Upon return, the AIF updates the first field to denote, in `AIF`*nn*`GET` AIFs, the actual number of elements returned, and for `AIF`*nn*`PUT` AIFs, the number of elements where valid information is located. If information is truncated because not enough elements were provided to return all the valid information, the corresponding *itemstatus_array* element of an `AIF`*nn*`GET` call provides a warning.

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)

Caution

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

Error Management

While error checking is minimized in order to increase AIF performance, architected interfaces provide comprehensive error management.

Architected interfaces provide parameters that return information about the success or failure of the call. Each status parameter uses the data type `status_type` to return status information. Following are the types of status parameters provided by many operating system AIFs:

- *overall_status*
- *itemstatus_array*
- *ver_item_statuses*

Each `AIF`*nn*`GET` interface checks to make sure that the specified item exists on the system and that the caller is of sufficient privileged level. If the caller meets the access criteria, checking will be done to ensure 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.

Each `AIF`*nn*`PUT` interface checks to make sure that the specified item exists on the system and that the caller is of sufficient privileged level. In addition, whenever possible, values are range checked prior to insertion.

Status Data Type

The data type required for use with status parameters is `status_type` (also described in appendix B):

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

If no error is detected, a status parameter returns a zero. If an error is detected, the status variable returns a negative value.

If errors are detected, the 32-bit value returned must be evaluated as an array of two 16-bit integers. The leftmost 16 bits (first element), evaluated as a 16-bit integer, return a status number and the rightmost 16 bits, evaluated as a 16-bit integer, return the subsystem number.

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

The status number provides error information. Appendix A provides a complete list of the AIF status numbers and their meanings.

Note Status variables must be initialized to zero before calling the AIF.

Overall Status

AIFs use the parameter *overall_status* to indicate the status of the overall call. If an AIF call is successful, zero is returned in *overall_status*. If an error has occurred, a negative number is returned.

A positive number returned indicates the index of the last element in the *items_array* parameter that caused an error. There might be more errors, but you must check each element in *itemstatus_array* to locate additional errors.

Note Always initialize all elements of *overall_status* to zero before calling AIFs.

Item Status

The *itemstatus_array* parameter is an array of `status_type`. This array returns status information on each individual item located in the *item_array* parameter. There is a one-to-one correspondence between elements in *itemstatus_array* and elements in *item_array*. For example, the eighth element of *itemstatus_array* returns status information about the eighth element of *item_array*. The *itemstatus_array* parameter is available with all information access AIFs and some functionality access AIFs. A positive value returned in an *itemstatus_array* element indicates a warning condition associated with the item in the corresponding *item_array* element. A negative value returned in an *itemstatus_array* element indicates that there is an error associated with the item in the corresponding *item_array* element.

In addition, *itemstatus_array* can return a special error indicating that more elements in the array could have been accommodated.

Note Always initialize all elements of *itemstatus_array* to zero before calling AIFs.

Item Verification Status

There are three optional parameters available with `AIF`*nn*`PUT` AIFs:

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

These parameters are arrays used to verify that specific conditions are true before information updating proceeds. If an item value in the *ver_items* array does not match the corresponding item in the *item_array* returned by the previous `AIF`*nn*`GET` call, none of the `AIF`*nn*`PUT` operations succeed.

The *ver_item_statuses* is an array of `status_type`. This array returns status information concerning the success or failure of verification on each item specified in the *ver_item_nums* parameter and the data pointed to by the *ver_items* parameter. There is a one-to-one correspondence between elements in *ver_item_statuses* and elements in *ver_item_nums*. For example, the eighth element of *ver_item_statuses* returns status information about the eighth element of *ver_item_nums*.

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

If verification fails, *overall_status* contains an error status number -24 (Verification failed). You must scan each element in the *ver_item_statuses* array to determine which item failed verification.

Note

Always initialize all elements of *ver_item_statuses* to zero before calling AIFs.

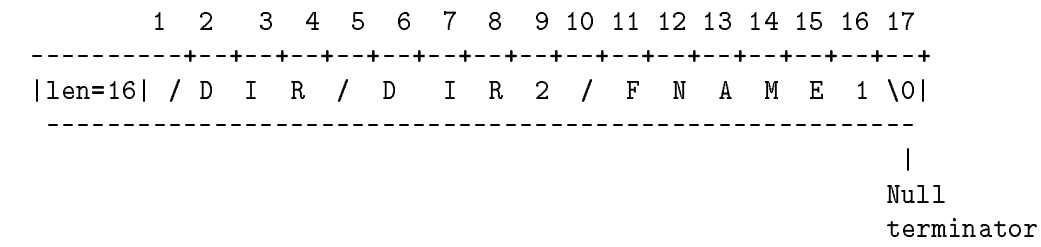
Hierarchical File System

MPE/iX Release 4.5 begins implementing features of POSIX. The largest impact that POSIX has on the Architected Interface Facility is the introduction of the Hierarchical File System (HFS). POSIX supports the specification of file pathnames that can reach a maximum path length of 1024 (including the null terminator).

This makes the concept of fixed size Return Arrays impractical for returning a list of pathnames since enormous arrays would need to be defined.

For those AIFs which return a list of filenames (for example, `AIFSYSWIDEGET` and `AIFPROCGET`), the names will be returned into a user buffer (the user may chose to pass in a long pointer to a mapped file, to a buffer in the user's stack, to an AIF global area). Each name will be terminated by a null character and the next name will follow starting in the next byte.

For those AIFs which return a single pathname, the user will specify on input the maximum pathname size in the first word of the user buffer. On output, the actual length of the pathname in bytes will be returned. All returned pathnames will be terminated with a null character which will not be included in the length. Following is an example of a pathname buffer.



If the pathname returned is too large to fit in the user buffer as specified by the length in the first word, then an error will be returned to the user application.

Note

For each AIF which accepts or returns a filename, new parameters or items have been added to support HFS pathnames. Existing items which are defined as MPE names (file.group.account) will not be effected. The idea is that eventually most applications will convert over to use the new HFS items, but they will not be forced to convert over immediately.

For more general information on POSIX, refer to *New Features of MPE/iX: Using the Hierarchical File System* (32650-90351).

3

Architected Interface Descriptions

This chapter describes operating system architected interfaces, arranged alphabetically.

AIFACCTGET

Returns system accounting information.

Syntax

```
                    REC             I32A            @64A
AIFACCTGET (overall_status, itemnum_array, item_array,
                    RECA                REC         I32
            itemstatus_array, directory_name, user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

directory_name **Record by reference (optional)**

Passes the directory name of the object for which information is desired. Group objects require group and account names. User objects require user and account names. Account objects require only the account name. If this parameter is not provided, the default user, group, and account names are used.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

In order to obtain information about a particular group you must specify both the group and account names, as the same group name can exist in multiple accounts (for example, `PUB`).

AIFACCTPUT

Modifies system accounting information.

Syntax

```
                    REC              I32A            @64A
AIFACCTPUT (overall_status, itemnum_array, item_array,
                    RECA             REC             I32
            itemstatus_array, directory_name, user_id,
                    I32A          @64A            RECA
            ver_item_nums, ver_items, ver_item_statuses);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

directory_name **Record by reference (optional)**

Passes the *directory_name* of the object for which information is desired. Group objects require group and account names. User objects require user and account names. Account objects require only the account name. If this is not provided, the default user, group, and account names are used.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element $n+1$ must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

Operation Notes

In order to modify information associated with a particular group, you must specify both the group and account names, as the same group name can exist in multiple accounts (for example, `PUB`).

AIFACCTGET/PUT Items

The following two tables provide summary and detailed descriptions of the item numbers associated with accounting information.

Item Summary

The following table summarizes the item numbers associated with accounting information. For more detailed information about these item numbers, refer to the table of accounting information item descriptions.

Table 3-1. Accounting Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
6001	CA16	User name	N	Y			
6002	CA16	User password	Y	Y			
6003	I32	User capabilities	Y	Y			
6004	I32	Maximum priority	Y	Y			
6005	I32	User logon count	N	Y			
6006	CA16	User home group	Y	Y			
6007	I32	User UDC Index	N	Y			
6008	I32	User local attributes	Y	Y			
6009	CA16	User password validation	N	N			
6010	pathname_type	Home directory	N	Y			
6011	I32	UID	N	Y			
6012	pathname_type	Initial Logon Program	N	Y			
6013	B	Encrypted	N	Y			
6014	B	User password required	N	Y			
6015	B	User password warning	N	Y			
6016	B	User password expired	N	Y			
6017	B	User password invalid	N	Y			
6018	B	User name invalid	N	Y			
6019	I32	Invalid user logon count	N	Y			
6020	U32	User password aging start date	N	Y			
6021	I32	User password aging minimum days	N	Y			
6022	I32	User password aging maximum days	N	Y			
6023	I32	Password aging warning days	N	Y			
6024	I32	Password aging expiration days	N	Y			
6025	CA16	Encrypted user password	N	N			
6101	CA16	Group name	N	Y			
6102	CA16	Group password	Y	Y			
6103	I32	Group capabilities	Y	Y			
6104	I32	Group access/security	Y	Y			
6105	I32	Group accumulated space	Y	Y			
6106	I32	Group maximum allowed space	Y	Y			

Table 3-1. Accounting Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
6107	I32	Group accumulated CPU time	Y	Y			
6108	I32	Group maximum allowed CPU time	Y	Y			
6109	I32	Group accumulated connect time	Y	Y			
6110	I32	Group maximum allowed connect time	Y	Y			
6111	I32	Linkage	N	Y			
6112	CA32	Volume set name	N	Y			
6113	CA16	Group password validation	N	N			
6114	B	Encrypted	N	Y			
6115	CA16	Encrypted group password	N	N			
6201	CA16	Account name	N	Y			
6202	CA16	Account password	Y	Y			
6203	I32	Account capabilities	Y	Y			
6204	I32	Account access/security	Y	Y			
6205	I32	Account accumulated space	Y	Y			
6206	I32	Account maximum allowed space	Y	Y			
6207	I32	Account accumulated CPU time	Y	Y			
6208	I32	Account maximum allowed CPU time	Y	Y			
6209	I32	Account accumulated connect time	Y	Y			
6210	I32	Account maximum allowed connect time	Y	Y			
6211	I32	Account maximum priority	Y	Y			
6212	I32	Account UDC index	N	Y			
6213	I32	SYS UDC index	N	Y			
6214	I32	Account local attributes	Y	Y			
6215	CA16	Account password validation	N	N			
6216	I32	GID	N	Y			
6217	B	Encrypted	N	Y			
6218	B	Account user passwords required	N	Y			
6219	CA16	Encrypted account password	N	N			

Note

Spaces in the columns for Min, Max, and Error# indicate that there are no current values for those items.

Item Descriptions

The following three tables provide detailed descriptions of item numbers and corresponding items associated with user, group, and account information.

Table 3-2. Accounting Information: User Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6001	**User name (CA16) Put: No; Verify: Yes; Release 3.0** Returns the user name (left-justified and padded with blanks).
6002	**Password (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the password of the specified user. The HP 3000 Security Monitor/iX Product available on Release 5.0 provides password encryption. With password encryption turned on, a new password is automatically encrypted before it is stored in the system directory. The encryption facility works one way. Even if you know the encryption algorithm, you cannot reconstruct a password in plain language from its encrypted version. Therefore, encrypted passwords are NOT returned in this item; instead the text "*ENCRYPTED*" is returned. See item 6025.
6003	**Capabilities (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the capability mask for this user. The item is a bitmap and, if the bit is set to 1, the user owns the corresponding capability. The user capabilities cannot be greater than the corresponding account capabilities. Bit (0:1) SM Bit (1:1) AM Bit (2:1) AL Bit (3:1) GL Bit (4:1) DI Bit (5:1) OP Bit (6:1) CV Bit (7:1) UV Bit (8:1) LG Bit (9:1) SP Bit (10:1) PS Bit (11:1) NA Bit (12:1) NM Bit (13:1) CS Bit (14:1) ND Bit (15:1) SF Bits (16:7) Unused (set to 0) Bit (23:1) BA Bit (24:1) IA Bit (25:1) PM Bits (26:2) Unused (set to 0) Bit (28:1) MR Bit (29:1) Unused (set to 0) Bit (30:1) DS Bit (31:1) PH

Table 3-2. Accounting Information: User Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6004	**Maximum priority (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the priority that is the maximum allowed for the user. The maximum priority for a user is specified by using the `MAXPRI` parameter of the `NEWUSER` and `ALTUSER` commands. The values and their associated queues are listed below: 100 BS queue 150 CS queue 200 DS queue 250 ES queue
6005	**Logon count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of jobs and/or sessions open for this user.
6006	**Home group (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the name of the home group of this user (left-justified and padded with blanks).
6007	**User UDC Index (I32) Put: No; Verify: Yes; Release 5.0** The offset into COMMAND.PUB.SYS for user UDCs. COMMAND.PUB.SYS reflects the UDC environment that takes effect the next time the user logs on.
6008	**Local attributes (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the user definable attributes of this user.
6009	**Password validation (CA16) Put: No; Verify: No; Release 3.0** Passes a user password. The corresponding status in the *itemstatus_array* will contain an error and the *overall_status* an index if the actual user password does not match the specified user password.
6010	**Home directory (REC) Put: No; Verify: Yes; Release 4.5** Returns the home directory associated with this user. The directory is in the format of an HFS pathname (for example, /SYS/PUB). Record type: `pathname_type` (Refer to appendix B.)
6011	**UID (I32) Put: No; Verify: Yes; Release 4.5** Returns the user ID associated with this user. The user ID (UID) provides file owner class security for MPE/iX. The UID is added to the user database, HPUID.PUB.SYS, when a new user is created with the `NEWUSER` command.
6012	**Initial Logon Program (REC) Put: No; Verify: Yes; Release 4.5** Returns the initial logon program for this user. The program will be represented as an HFS pathname (for example, /SYS/PUB/CI). Record type: `pathname_type` (Refer to appendix B.)
6013	**Encrypted? (B) Put: No; Verify: Yes; Release 5.0** Returns true if the user password is encrypted and false when the user password is plain text. The encryption feature is enabled in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.

Table 3-2. Accounting Information: User Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6014	**User Password Required? (B) Put: No; Verify: Yes; Release 5.0** Returns true if the user password is required and false when it is not. The required password is set via the USERPASS=REQ option on the `NEWACCT` and `ALTACCT` commands when the HP Security Monitor is installed. For more information see the *MPE/iX Security Features System Manager's Guide*.
6015	**User Password Warning? (B) Put: No; Verify: Yes; Release 5.0** Returns true if the user password warning is in effect and false when it is not. During the password warning period, the logon user is notified of the pending password expiration. This feature is enabled by the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6016	**User Password Expired? (B) Put: No; Verify: Yes; Release 5.0** Returns true if the user password is expired and false when it is not. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6017	**User Password Invalid? (B) Put: No; Verify: Yes; Release 5.0** Returns true if the user password is invalid and false when it is not. An invalid user password is one which has exceeded the maximum lifetime or expiration period. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6018	**User Name Invalid? (B) Put: No; Verify: Yes; Release 5.0** Returns true if the user name (ID) is invalid. A user name becomes invalid when the invalid user logon count has been exceeded. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6019	**Invalid User Logon Count (I32) Put: No; Verify: Yes; Release 5.0** Returns the number of invalid logon attempts for a user name. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6020	**User Password Aging Start Date (U32) Put: No; Verify: Yes; Release 5.0** Returns the start date of the password aging cycle. Password aging is enforced only on REQUIRED user passwords. This value is used only when item 6022 (maximum days) is greater than zero. The bits and their meanings are as follows: Bits (0:16) Unused Bits (16:7) The year of the century Bits (23:9) The day of the year This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.

Table 3-2. Accounting Information: User Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6021	**User Password Aging Minimum Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the minimum period in days a new or changed user password cannot be altered. Password aging is enforced only on REQUIRED user passwords. This value is used only when item 6022 (maximum days) is greater than zero. A valid range of values is 0 to 364 days. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6022	**User Password Aging Maximum Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the maximum period in days for which a user password is valid, this includes the expiration period. Password aging is enforced only on REQUIRED user passwords. When the user maximum is set, the global user maximum age is ignored. The user maximum days cannot exceed the global user maximum days. A valid range of values is 0 to 365 days. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6023	**Password Aging Warning Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the warning period in days the user is given before the password expires. Password aging is enforced only on REQUIRED user passwords. This value is used only when item 6022 (maximum days) is greater than zero. A valid range of values is 0 to 364 days. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6024	**Password Aging Expiration Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the number of days a user password is expired before becoming invalid. A user can still change an expired password. Once the password exceeds the expired period it is placed in an invalid state. Once invalid, only the system or account manager can change the password. Password aging is enforced only on REQUIRED user passwords. A valid range of values is 0 to 364 days. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6025	**Encrypted Password (CA16) Put: No; Verify: No; Release 5.0** Returns the encrypted password of the specified user if one exists, otherwise, blanks are returned. See item 6002.

Table 3-3. Accounting Information: Group Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6101	**Group name (CA16) Put: No; Verify: Yes; Release 3.0** Returns the name of the group (left-justified and padded with blanks).
6102	**Password (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the password of the specified group. The HP 3000 Security Monitor/iX Product available on Release 5.0 provides password encryption. With password encryption turned on, a new password is automatically encrypted before it is stored in the system directory. The encryption facility works one way. Even if you know the encryption algorithm, you cannot reconstruct a password in plain language from its encrypted version. Therefore, encrypted passwords will NOT be returned in this item; instead the text "*ENCRYPTED*" is returned. See item 6015.
6103	**Capabilities (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the capability mask for this group. It is a bit map and, if the bit is set to 1, the group owns the corresponding capability. The group capabilities cannot be greater than the corresponding account capabilities. Bits and their meanings are listed below: Bits (0:23) Unused (set to zero) Bit (23:1) BA Bit (24:1) IA Bit (25:1) PM Bits (26:2) Unused (set to zero) Bit (28:1) MR Bits (29:1) Unused (set to zero) Bit (30:1) DS Bit (31:1) PH

Table 3-3. Accounting Information: Group Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6104	**Access (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the group access/security mask. Bits and their meanings are: Bits (0:2) Unused Bit (2:1) Read any Bit (3:1) Read account user Bit (4:1) Read account librarian Bit (5:1) Read group user Bit (6:1) Read group librarian Bit (7:1) Append any Bit (8:1) Append account user Bit (9:1) Append account librarian Bit (10:1) Append group user Bit (11:1) Append group librarian Bit (12:1) Write any Bit (13:1) Write account user Bit (14:1) Write account librarian Bit (15:1) Write group user Bit (16:1) Write group librarian Bit (17:1) Lock any Bit (18:1) Lock account user Bit (19:1) Lock account librarian Bit (20:1) Lock group user Bit (21:1) Lock group librarian Bit (22:1) Execute any Bit (23:1) Execute account user Bit (24:1) Execute account librarian Bit (25:1) Execute group user Bit (26:1) Execute group librarian Bit (27:1) Save any Bit (28:1) Save account user Bit (29:1) Save account librarian Bit (30:1) Save group user Bit (31:1) Save group librarian
6105	**Accumulated space (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of sectors currently allocated to files in this group.
6106	**Maximum space (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of sectors that may be allocated to files in this group.
6107	**Accumulated CPU (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of CPU seconds used by this group.

Table 3-3. Accounting Information: Group Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6108	**Maximum CPU (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum amount of CPU seconds allowed for this group.
6109	**Accumulated connect (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the accumulated connect time in minutes for this group.
6110	**Maximum connect (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of connect minutes allowed for this group.
6111	**Linkage (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of accounts this group is linked into. Currently has a value of 1.
6112	**Volume set name (CA32) Put: No; Verify: Yes; Release 3.0** Returns the name of the volume set on which this group resides (left-justified and padded with blanks).
6113	**Password validation (CA16) Put: No; Verify: No; Release 3.0** Passes a group password. The corresponding status in the *itemstatus_array* will contain an error and the *overall_status* an index if the actual group password does not match the specified group password.
6114	**Encrypted? (B) Put: No; Verify: No; Release 5.0** Returns true if the group password is encrypted and false when the group password is plain text. The encryption feature is enabled in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6115	**Encrypted Password (CA16) Put: No; Verify: No; Release 5.0** Returns the encrypted password of the specified group if one exists, otherwise, blanks are returned. See item 6102.

Table 3-4. Accounting Information: Account Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6201	**Account name (CA16) Put: No; Verify: Yes; Release 3.0** Returns the account name (left-justified and padded with blanks).
6202	**Password (CA8) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the password of the specified account. The HP 3000 Security Monitor/iX Product available on Release 5.0 provides password encryption. With password encryption turned on, a new password is automatically encrypted before it is stored in the system directory. The encryption facility works one way. Even if you know the encryption algorithm, you cannot reconstruct a password in plain language from its encrypted version. Therefore, encrypted passwords are NOT returned in this item; instead the text "*ENCRYPTED*" is returned. See item 6219.
6203	**Capabilities (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the capability mask for this account. It is a bit map and, if the bit is set to 1, the account owns the corresponding capability. The account capabilities cannot be less than the corresponding user and group capabilities. Bits and their meanings are listed below: Bit (0:1) SM Bit (1:1) AM Bit (2:1) AL Bit (3:1) GL Bit (4:1) DI Bit (5:1) OP Bit (6:1) CV Bit (7:1) UV Bit (8:1) LG Bit (9:1) SP Bit (10:1) PS Bit (11:1) NA Bit (12:1) NM Bit (13:1) CS Bit (14:1) ND Bit (15:1) SF Bits (16:7) Unused (set to zero) Bit (23:1) BA Bit (24:1) IA Bit (25:1) PM Bits (26:2) Unused (set to zero) Bit (28:1) MR Bit (29:1) Unused (set to zero) Bit (30:1) DS Bit (31:1) PH

Table 3-4. Accounting Information: Account Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6204	**Access (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the account access/security mask: Bit (0:4) Unused (set to zero) Bit (4:1) Read any Bit (5:1) Read AC Bit (6:1) Append any Bit (7:1) Append AC Bit (8:1) Write any Bit (9:1) Write AC Bit (10:1) Lock any Bit (11:1) Lock AC Bit (12:1) Execute any Bit (13:1) Execute AC Bit (14:1) Save any Bit (15:1) Save AC Bit (16:16) Unused (set to zero)
6205	**Accumulated space (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of sectors currently allocated to files in this account.
6206	**Maximum space (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of sectors that may be allocated to files in this account.
6207	**Accumulated CPU (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of CPU seconds used by this account.
6208	**Maximum CPU (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of CPU seconds allowed for this account.
6209	**Accumulated connect (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the accumulated connect time in minutes for this account.
6210	**Maximum connect (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of connect minutes allowed for this account.
6211	**Maximum priority (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies a priority that is the maximum allowed for the account. The maximum priority for an account is specified by using the `MAXPRI` parameter of the `NEWACCT` and `ALTACCT` command. The values and their associated queues are: 100 BS queue 150 CS queue 200 DS queue 250 ES queue

Table 3-4. Accounting Information: Account Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
6212	**Account UDC Index (I32) Put: No; Verify: Yes; Release 5.0** The offset into COMMAND.PUB.SYS for account UDCs. COMMAND.PUB.SYS reflects the UDC environment that takes effect the next time the user logs on.
6213	**System UDC Index (I32) Put: No; Verify: Yes; Release 5.0** The offset into COMMAND.PUB.SYS for system UDCs. COMMAND.PUB.SYS reflects the UDC environment that takes effect the next time the user logs on.
6214	**Local attributes (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the user definable attributes of this account.
6215	**Password validation (CA16) Put: No; Verify: No; Release 3.0** Passes an account password. The corresponding status in the *itemstatus_array* will contain an error and the *overall_status* an index if the password does not match the specified account password.
6216	**GID (I32) Put: No; Verify: No; Release 4.5** Returns the Group ID associated with this account. The Group ID (GID) provides file group class security for MPE/iX. The GID is added to the group database, HPGID.PUB.SYS, when a new account is created with the NEWACCT command.
6217	**Encrypted? (B) Put: No; Verify: No; Release 5.0** Returns true if the account password is encrypted and false when the account password is plain text. The encryption feature is enabled in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
6218	**Account Users Passwords Required (B) Put: No; Verify: No; Release 5.0** Returns true when all users in an account are required to have user level passwords. The required password is set via the USERPASS=REQ option on the `NEWACCT` and `ALTACCT` commands when the HP Security Monitor is installed. For more information see the *MPE/iX Security Features System Manager's Guide*.
6219	**Encrypted Password (CA16) Put: No; Verify: No; Release 5.0** Returns the encrypted password of the specified account if one exists, otherwise, blanks are returned. See item 6202.

AIFCHANGELOGON

Changes the logon environment of a process.

Syntax

```
                          REC          CA          REC       I32
AIFCHANGELOGON (overall_status, logon_cmd, logon_desc, options,
                        REC       I32
                  error_status, user_id);
```

Parameters

overall_status **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 : `status_type` (Refer to appendix B.)

logon_cmd **character array by reference (optional)**

logon_cmd must be declared as a packed array less than or equal to 128 characters in length, and terminated by either a NULL character (ASCII 0) or a carriage return (ASCII 13).

The format of *logon_cmd* is:

jobname, ***user/userpass.acct/acctpass***, *group/grouppass*

The parameters ***userpass***, ***acctpass***, and *grouppass* refer to the user, account, and group passwords, respectively. The *jobname* and *group/grouppass* portions of *logon_cmd* are optional. The default is that no jobname is assigned. The default for group is your home group if you are assigned one by the account manager. This parameter is required if a home group is not assigned to user.account.

If *logon_cmd* is passed, *logon_desc* can be passed to return the target logon environment (including the home group name) in the `logon_desc_type` format (refer to appendix B.) You must pass either *logon_cmd* or *logon_desc* or both.

Default: nil

logon_desc **record by reference (optional)**

Required if *logon_cmd* is not passed. Passes the target logon environment in a variable declared as a `logon_desc_type`. If the group is not specified in the `group_name` field, the target user.account's home group is returned in that field.

If *logon_cmd* is passed, *logon_desc* can be passed to return the target logon environment (including the home group name) in the `logon_desc_type` format. (Refer to appendix B.)

You must pass either *logon_cmd* or *logon_desc* or both.

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

Default: nil

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

Directs `AIFCHANGELOGON` to skip some of the usual steps performed in changing the logon environment. Following are the bit definitions corresponding to the various options (set the bit to 1 to invoke the option, all the other bits should be set to zero):

Bit (0:1) Do not change the global job name (listed when you use the `SHOWJOB` command). When this bit is set, only the process local job name is updated. The global (jobwide or sessionwide) job name remains unchanged. For example, the `SHOWME` command displays the new job name of the local process, and the `SHOWJOB` command displays the original job name (the same one that would have been displayed before the `AIFCHANGELOGON`).

Bit (1:1) Do not change the global user and account name. When this bit is set, only the process local user and account names are updated. The global (jobwide or sessionwide) user and account names remain unchanged. For example, the `SHOWME` command

	displays the new user and account names of the local process, and the SHOWJOB command displays the original logon user and account names (the same one that would have been displayed before the AIFCHANGELOGON).
Bit (2:1)	Do not change the global group name. When this bit is set, only the process local group name is updated. The global (jobwide or sessionwide) user and account name remains unchanged. For example, the SHOWME command displays the new group name of the local process and the SHOWJOB command displays the original logon group name (the same one that would have been displayed before the AIFCHANGELOGON).
Bit (3:1)	Do not change the allow mask.
Bit (4:1)	Keep the current temporary file directory. If this bit is not set and the process has files open, AIFCHANGELOGON returns an error.
Bit (5:1)	Keep current file equations. If this bit is not set, after an AIFCHANGELOGON all of the file equations issued prior to calling AIFCHANGELOGON are reset.
Bit (6:1)	Not used. Set to zero.
Bit (7:1)	Do not perform password validation.
Bit (8:24)	Reserved. Set to zero.
Default: 0	

error_status **record by reference (optional)**

Returns a valid error number only if -2510 is returned in the `info` field of ***overall_status***, indicating that the target logon environment passed in *logon_cmd* is not syntactically valid. You can pass *error_status* to the `HPERRMSG` intrinsic to return a syntax error message.

Refer to the *MPE/iX Intrinsics Reference Manual* (32650-90028) for a description of `HPERRMSG`.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

The `AIFCHANGELOGON` AIF changes the logon environment of a process. It supports the concept of a private logon environment, so the effects of `AIFCHANGELOGON` are local to the process. This allows users to call `AIFCHANGELOGON` from multiple processes executing within a given job/session without having the various processes interfere with each other. All child processes created after calling `AIFCHANGELOGON` inherit the user name, account name, group name, job name, and capabilities of the parent. Processes created prior to calling `AIFCHANGELOGON` are not affected.

Any program which has called `AIFCHANGELOGON` and has used the options parameter to change the global logon environment must call `AIFCHANGELOGON` again to restore the logon environment to its original state before terminating. It the global logon environment is not restored, the parent process might experience difficulties when accessing logon related information and at the time of logoff.

Operation Notes - Current Restrictions

The current implementation of this procedure is subject to the following restrictions:

Session Variables:

There is only one variable table per job or session. Session variables, both user-defined and system-defined, are stored by variable name in this table. If multiple processes are executing in the same job/session, they all share the same variables. If one process issues a programmatic `SETVAR` command and another process issues

a programmatic DELETEVAR or SETVAR command for the same variable name, the SETVAR issued by the first process is deleted or overwritten. The AIFCHANGELOGON AIF does not create private (process-local) variables.

System Variables

Most system variables (HP@) are actually implemented as "active functions", and they function correctly after a process executes an AIFCHANGELOGON. They should reflect the changes for the process. A few system variables are not implemented as active functions. These system variables will experience the same behavior as user-defined variables; one process can overwrite the changes made by another process in the same job/session.

Below is a complete list of system variables implemented as active functions. The variables marked with an "*" are read/write variables; the rest are read only.

```
 HPACCOUNT      HPACCTCAP     HPACCTCAPF    *HPAUTOCONT    HPCIDEPTH
 HPCIERRMSG     HPCMDNUM     *HPCMDTRACE     HPCONNMINS    HPCONNSECS
 HPCONSOLE      HPCONTINUE    HPCPUMSECS     HPCPUNAME     HPCPUSECS
 HPDATE         HPDATEF       HPDAY          HPDTCPORTID   HPDUPLICATIVE
*HPERRDUMP     *HPERRSTOLIST  HPEXECJOBS     HPGROUP       HPGROUPCAP
 HPGROUPCAPF    HPHGROUP      HPHOUR         HPINBREAK     HPINPRI
 HPINTERACTIVE  HPINTRODATE   HPINTROTIME    HPJOBCOUNT    HPJOBLIMIT
 HPJOBFENCE     HPJOBNAME     HPJOBNUM       HPJOBTYPE     HPLDEVIN
 HPLDEVLIST     HPMINUTE      HPMONTH       *HPMSGFENCE    HPNCOPIES
 HPOUTCLASS     HPOUTFENCE    HPQUIET       *HPREDOSIZE    HPSCHEDJOBS
 HPSESCOUNT     HPSESLIMIT    HPSTDIN        HPSTDLIST     HPSUSAN
 HPSUSPJOBS     HPTIMEF      *HPTIMEOUT     *HPTYPEAHEAD   HPUSER
 HPUSERCAP      HPUSERCAPF    HPUSERCMDEPTH  HPUSERSCOUNT  HPUSERLIMIT
 HPVERSION      HPWAITJOBS    HPYEAR
```

Temporary Files

The default for AIFCHANGELOGON is to create a new temporary directory on release 4.0. For applications which had temporary files open this resulted in errors being returned. In the past, the temporary directory was shared by all processes in the job/session domain. Unless the application has a need to create a new temporary directory, the recommendation is to set bit 4 in the options parameter to keep the existing temporary directory. When bit 4 is not set, the caller of AIFCHANGELOGON must close all temporary files. If temporary files are not closed, and the option to keep the temporary directory is not set, then AIFCHANGELOGON returns an error.

JOBINFO

If a process calls AIFCHANGELOGON, then information about the process local logon environment (created my AIFCHANGELOGON) will not be accessible via the JOBINFO intrinsic. The information returned by JOBINFO always reflects the global (jobwide or sessionwide) logon environment. If options to update global information are not selected, the global information is going to be different from the process local information. To avoid confusion and assure consistency use AIFJSGET/PUT and AIFPROCGET/PUT.

DSCOPY

The DSCOPY command does not work correctly when invoked programmatically from a process that has changed its logon environment using AIFCHANGELOGON. The DSCOPY process inherits the original logon characteristics instead of the process local environment. As a result, the capabilities of the DSCOPY process may be different (more or less).

DSCOPY capabilities problem

If the original capability is a superset of the new capability, DSCOPY grants access to files that the process should not have access to. On the other hand, if the original capability is less (not a superset) then the new capabilities, DSCOPY denies access to files that the process should have access to.

DSCOPY non-fully qualified problem

Suppose that you change logon to a new group or account, and you do a DSCOPY as follows:

```
DSCOPY filename[.groupname[.acctname]]
```

If *groupname* is omitted, the file system qualifies the group name with your original logon group name. Similarly, if *acctname* is omitted, the file system qualifies the account name with your original logon account name.

UDC environment

The AIFCHANGELOGON AIF does not execute the logon UDC as a regular logoff and logon would. The UDC environment stays the same as the original logon. The new user may not be able to use the original logon UDC anymore if he or she does not have the right capabilities.

AIFCLOSE

Allows files to be saved across account boundaries.

Syntax

```
                REC          I16          I16        I16      I32
AIFCLOSE (overall_status, file_number, disposition, sec_code, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

file_number **16-bit signed integer by value (required)**

Passes the file number of the file to be closed.

disposition **16-bit signed integer by value (required)**

Passes the disposition of the file, valid only for files on disk and magnetic tape. This disposition can be overridden by a file equation.

The disposition options are defined as follows:

Bits (0:12) Reserved for MPE/iX. Set to zero.

Bit (12:1) Disk Space

- 0 Do not return disk space beyond file EOF.
- 1 Return disk space beyond file EOF.

Bits (13:3) File domain

- 000 No change
- 001 Permanent file
- 010 Temporary file (rewound)
- 011 Temporary file (not rewound)
- 100 Released file

Refer to the description of the `FCLOSE` intrinsic in the *MPE/iX Intrinsics Reference Manual* (32650-90028) for more information about this parameter.

sec_code **16-bit signed integer by value (required)**

Passes the type of security initially applied for new permanent files.

0 Unrestricted access.
1 Private file creator security.

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

If `AIFCLOSE` fails, either a bad file number was specified, another file with the same name already exists, an illegal disposition (5,6,7) was passed, or any outstanding write I/Os may have failed. Use `FCHECK` to determine the reason `AIFCLOSE` failed.

AIFCONVADDR

Converts compatibility mode relative addresses to the corresponding native mode virtual addresses.

Syntax

```
                    REC           I32A          RECA
AIFCONVADDR (overall_status, mode_array, inaddress_array,
                    @64A            I32A          I32
          outaddress_array, convstatus_array, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

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

Passes an array of integers where each element contains a value indicating the addressing mode of the compatibility mode address located in the corresponding element in ***inaddress_array***. If n addresses are being converted, element $n + 1$ must be a zero to indicate the end of the element list.

Values and their meanings are as follows:

1	DB relative byte address (stack or XDS)
2	DB relative word address (stack or XDS)
3	Stack DB relative byte address (stack only)
4	Stack DB relative word address (stack only)
5	Bank 0 relative word address

inaddress_array **record array by reference (required)**

An array where each element is an unsigned 16-bit CM address to be converted. The addressing mode of the CM address is defined in the corresponding element in ***mode_array***.

Array type: `bit16` (Refer to appendix B.)

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

An array where each element returns a 64-bit address that is the result of the conversion performed on a CM address located in the corresponding element in ***inaddress_array***.

Array type: `globalanyptr`

convstatus_array **record array by reference (required)**

An array where each element returns the status of the conversion operation performed in the corresponding element in ***inaddress_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

Array 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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

None.

AIFDEVCLASSGET

Returns information for a device class.

Syntax

```
                              REC          I32A        @64A
AIFDEVCLASSGET (overall_status, itemnum_array, item_array,
                              RECA         CA16          I32
                itemstatus_array, device_class, device_class_key,
                  I32
                user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

device_class **16-byte character array by reference (optional)**

The user configured device class name for the request. The name must be capitalized and blank filled.

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

This is the index to the device class table. This is the fast key to retrieve the device class information.

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

The `AIFDEVCLASSGET` call requires *device_class* or *device_class_key* as an input parameter. Both search keys can be obtained by calling the `AIFSYSWIDEGET` procedure area 13500.

If both *device_class* name and *device_class_key* are provided, the values are checked against each other. If the two keys do not identify the same class, `AIFDEVCLASSGET` terminates with an error condition.

AIFDEVCLASSGET Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with device criteria items used by AIFDEVCLASSGET.

Table 3-5. AIFDEVCLASSGET - Device Criteria Items from System Tables

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description

13501	**Devices (Record) Get: Yes; Put: No; Verify: Yes; Release 4.0**
	This item returns the LDEVs in the device class. The format of the record is as follows:
	`record`
	`  size  :integer`
	`  ldev_array:array[1..size]of integer;`
	`end`
	The first word of the record holds the number of ldev's retrieved and the rest of the record holds the ldev's.

13502	**User-defined Device Class Name (C16) Get: Yes; Put: No; Verify: Yes; Release 4.0** This item returns the device class name assigned to the device by the user.
13503	**Device Class Key (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the class index to the Device Class Table.
13504	**Number of Devices in Class (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Number of devices in the device class.
13505	**Device Class Access Type (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the device class access type. 0-7 Disk 16-23 Terminal 24-31 Tape 32-37 Printer

AIFDEVICEGET

Returns characteristics for devices.

Syntax

```
                      REC          I32A          I64A
AIFDEVICEGET (overall_status, itemnum_array, item_array,

                      I32A        I32       REC
                itemstatus_array, ldev,  device_key,

                  I32
                user_id) ;
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

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

The logical device number for the request.

device_key **record by reference (optional)**

The *ufid* for the device file.

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

The `AIFDEVICEGET` call requires *ldev* or *device_key*, which specify the requested device. Both *ldev* and *device_key* can be obtained by calling the `AIFSYSWIDEGET` procedure area 13000. Items are divided into four classes: system tables, terminals, disks, and tape drives. In most cases, the device must be configured in order to return item values.

All device control functions (for items 13100-13399) are queued and processed serially. For a device with an outstanding I/O request, the control function request from the AIF device call is not processed until the pending I/O is complete.

If the device is a terminal, the AIF call is processed after either the pending read is complete or the terminal read times out.

The AIF device control terminal functions (13101-13138) do not guarantee the behavior is the same for all types of terminal connections. The behavior is highly dependent on the low-level drivers and type of connection. The items are not guaranteed to work for all connection types since the functionality may not be supported in the lower level drivers. For more information on the behavior of the various terminal connections see the *Asynchronous Serial Communications Programmer's Reference Manual* (32022-61001).

The AIFs perform many of the same operations as FCONTROL and FDEVICECONTROL. They do not provide capabilities that are not already available through these interfaces. On MPE/iX Release 5.0 there are performance improvements to the FCONTROL and FDEVICECONTROL paths for DTC terminal connections that are not available through the I/O interfaces used by the AIFs. Therefore, the recommendation is to use the DTC terminal control functions provided by FCONTROL and FDEVICECONTROL on Release 5.0 and later.

Many of the device control functions can only occur while the device is open. If the caller does not have the device open, it will be opened on the caller's behalf. However, the device opened by the AIF is closed before returning to the caller. This often returns the device to the settings it had prior to the open. For the device to maintain the settings it should be opened and closed by the caller.

AIFDEVICEPUT

Modifies device characteristics or performs various control operations on configured devices.

Syntax

```
                    REC              I32A         @64A
AIFDEVICEPUT (overall_status, itemnum_array,   item_array,

                    I32A         I32        REC
              itemstatus_array, ldev,   device_key,

                 I32        I32A          @64A
              user_id, ver_item_nums,   ver_items,

                    I32
              ver_item_statuses);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

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

The logical device number for the request.

device_key **record by reference (optional)**

The *ufid* for the device file.

Record type: `ufid_type`.

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element $n+1$ must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

Operation Notes

The `AIFDEVICEPUT` call requires *ldev* or *device_key*, which specify the requested device. Both *ldev* and *device_key* can be obtained by calling the `AIFSYSWIDEGET` procedure area 13000. Items are divided into four classes: system tables, terminals, disks, and tape drives. In most cases, the device must be configured in order to return item values.

All device control functions (for items 13100-13399) are queued and processed serially. For a device with an outstanding I/O request, the control function request from the AIF device call is not processed until the pending I/O is complete.

If the device is a terminal, the AIF call is processed after either the pending read is complete or the terminal read times out.

The AIF device control terminal functions (13101-13138) do not guarantee the behavior is the same for all types of terminal connections. The behavior is highly dependent on the low-level drivers and type of connection. The items are not guaranteed to work for all connection types since the functionality may not be supported in the lower level drivers. For more information on the behavior of the various terminal connections see the *Asynchronous Serial Communications Programmer's Reference Manual* (32022-61001).

The AIFs perform many of the same operations as FCONTROL and FDEVICECONTROL. They do not provide capabilities that are not already available through these interfaces. On MPE/iX Release 5.0 there are performance improvements to the FCONTROL and FDEVICECONTROL paths for DTC terminal connections that are not available through the I/O interfaces used by the AIFs. Therefore, the recommendation is to use the DTC terminal control functions provided by FCONTROL and FDEVICECONTROL on Release 5.0 and later.

Many of the device control functions can only occur while the device is open. If the caller does not have the device open, it will be opened on the caller's behalf. However, the device opened by the AIF is closed before returning to the caller. This often returns the device to the settings it had prior to the open. For the device to maintain the desired settings it should be opened and closed by the caller.

AIFDEVICEGET/PUT Items

The following tables provide detailed descriptions of item numbers and corresponding items used by `AIFDEVICEGET` and `AIFDEVICEPUT`.

Device Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with device criteria items used by AIFDEVICEGET and AIFDEVICEPUT.

Table 3-6. AIFDEVICEGET/PUT - Device Criteria Items from System Tables

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13001	**LDEV (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** This is the LDEV for the device.
13003	**Device Type (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** This is the type of device as shown below: 0-7 Disk 16 Terminal 24 Tape 32 Printer Devices are recognized by MPE/iX through the system configuration software. For more information on device types, refer to the system configuration file IODFAULT.PUB.SYS or to the I/O configurator command LDEV in SYSGEN.
13004	**Device Subtype (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** This is the device subtype. For more information on device subtypes, refer to the system configuration file IODFAULT.PUB.SYS or to the I/O configurator command LDEV in SYSGEN.
13005	**JSMAIN PIN (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the JSMAIN PIN for the owner of the device.
13006	**User-Defined Device Name (CA16) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the real device name given with the ADEV option in SYSGEN.
13007	**Alternate Owner PIN (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the alternate owner PIN of the specified device.
13008	**Auto Reply (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** True if the device will automatically reply to tape requests (also called auto-allocation).
13009	**Job Accepting (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** True if the device accepts HELLO and JOB logons.
13010	**Data Accepting (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** True when the device accepts DATA logons.
13011	**Duplicative (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** True when all input operations for a job or session are echoed to a corresponding device without intervention by the operating system software.

Table 3-6.
AIFDEVICEGET/PUT - Device Criteria Items from System Tables (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13012	**BOT (B) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns true when the tape is at Load Point (beginning of tape); otherwise it returns false.
13013	**Interactive (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** True when the device requires human intervention for all input operations. This is necessary to establish the person/machine dialog required to support a session.
13014	**Record Width (I32) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** Returns or modifies the record width of this device.
13015	**Spool State (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the device spool state as follows: 0 Not spooled 1 Owned by an input spooler 2 Owned by an output spooler
13016	**Device Ownership State (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** 0 Not owned by any process 1 Owned by a process 2 The operating system has temporarily reserved the device 3 The operating system has temporarily reserved the device The states two and three are transitory states. Once complete the device should return to an owned or unowned state.
13017	**Device Is Up (B) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns true if the device is up. It returns false when the device is down.
13018	**Downed Request Pending (B) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns true if a down request is pending for the device; otherwise it returns false.
13019	**Trailer Disable (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** Returns true if the trailer pages are suppressed in printing; otherwise it returns false.
13020	**Header Disable (B) Get: Yes; Put: Yes; Verify: Yes; Release 4.0** Returns true if the header pages are suppressed in printing; otherwise it returns false.
13021	**Spool Queues are Open (B) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns true if the spooling has been enabled for the device; otherwise it returns false.
13022	**Special Forms Mounted (B) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns true if special forms are mounted; otherwise it returns false.

Table 3-6.
AIFDEVICEGET/PUT - Device Criteria Items from System Tables (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13023	**Formal File Name Designator (CA8) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the formal device file designator (for example, `$STDIN`). This name is left-justified and is padded with blanks to the right.
13024	**J/S Key (I32) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the job or session key, which can be passed to `AIFJSGET/PUT`.
13025	**I/O Device Class (I32) Get: Yes; Put: No; Verify: Yes; Release 4.5** Returns and verifies the system I/O device class. 0 Device not configured 1 Disk 2 Tape drive 3 Terminal 4 Printer (printers having a CIPER_DM logical device manager) 5 Printer (other non CIPER_DM printers listed in IODFAULT.PUB.SYS) 6 Spooled device 7 Data communication device 8 DS terminal 9 DS printer 10 User-defined device
13026	**I/O Device Subclass (I32) Get: Yes; Put: No; Verify: Yes; Release 4.5** Returns and verifies the system I/O device subclass. The subclass definition is based on the I/O device class listed in item 13025. For Terminals 0 Unknown 1 Device connected to a TMUX 2 Terminal 3 Printer 4 Virtual terminal 5 Virtual printer 6 PAD terminal 7 PAD printer 12 Null terminal 13 DHCF terminal 14 Pseudo terminal 15 Pseudo null terminal For Tape 1 Tape For Disk 1 Disk

Table 3-6.
AIFDEVICEGET/PUT - Device Criteria Items from System Tables (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13027	**Security Downed Device (B) Get: Yes; Put: No; Verify: Yes; Release 5.0** Returns true when the device has been downed by Security. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
13028	**Invalid Device Logon Count (I32) Get: Yes; Put: No; Verify: Yes; Release 5.0** Returns the current invalid logon count for the specified interactive device. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
13029	**Terminal Password? (B) Get: Yes; Put: No; Verify: Yes; Release 5.0** Returns true when the specified terminal has a password. Only nailed terminals can have passwords. This is a feature of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
13063	**Device Key (ufid_type) Get: Yes; Put: No; Verify: Yes; Release 4.0** Returns the *ufid* for the device file.

Terminal Device Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with terminal device items used by `AIFDEVICEGET` and `AIFDEVICEPUT`.

Table 3-7. AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13101	**Terminal Type (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the system-defined terminal type to be associated with an asynchronous port. 10 TT10 18 TT18 21 TT21 22 TT22 24 TT24 26 TT26 The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). Console device managers only support type 10. See `FCONTROL` (38) in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13102	**Line Speed (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the line speed setting for a terminal. All characters are 8 bits long (including the optional parity bit) plus two framing bits for a total of ten bits per character. The only supported line speeds are: 30 chars/sec 300 bits/sec 120 chars/sec 1200 bits/sec 240 chars/sec 2400 bits/sec 480 chars/sec 4800 bits/sec 960 chars/sec 9600 bits/sec 192 chars/sec 19200 bits/sec 3840 chars/sec 38400 bits/sec { direct connect devices on DTC 72MX only } The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). Not valid on the physical console. See `FCONTROL` (10,11,40) and `FDEVICECONTROL` 3 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13103	**Parity Enable (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the parity generation and checking. If disabled, all eight bits of each character are passed untouched by the driver to the device. If enabled, the "parity setting" determines what kind of parity checking/generation is in effect. Both input and output parity are the same. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). This item is not valid on the physical console. See `FCONTROL` (23,24) and `FDEVICECONTROL` 9,11 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.

Table 3-7.
AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13104	**Parity Setting (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the parity setting. 0 Forced to zero 1 Forced to one 2 Even 3 Odd 4 None The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). This item is not valid on the physical console. See `FCONTROL` (36) in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13105	**Echo enabled (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the echo status. When echo is enabled, all characters transmitted to the DTC are "echoed" back and appear on the terminal screen. In binary and block modes, this request is a no_op. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FCONTROL` (12,13) and `FDEVICECONTROL` 4 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13106	**Echo End of Record and Newline (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13107	**Additional End of Record (C) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13108	**Unedited Terminal Mode - EOR (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item allows the user to replace the EOR character in unedited (transparent) terminal mode. Unedited mode is nearly binary; an EOR, subsystem break character, and the AEORs are the only special characters. Unedited mode is enabled by using non-null EOR and subsystem break characters. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FCONTROL` (41) and `FDEVICECONTROL` 15 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.

Table 3-7.
AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13109	**Unedited Terminal Mode - Subsystem Break (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item allows the user to replace the subsystem break character in unedited (transparent) terminal mode. Unedited mode is nearly binary; an EOR, subsystem break character, and the AEORs are the only special characters. Unedited mode is enabled by using non-null EOR and subsystem break characters. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FCONTROL` (41) and `FDEVICECONTROL` 15 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13110	**Binary Edit Mode (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13111	**Block Mode Alert Character (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the HP block mode alert character. The normal alert character is DC2. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 29 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information. Not supported by console device managers.
13112	**Enable/Disable user block mode (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13113	**Enable/Disable VPLUS Block Mode (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13114	**Read Timeout (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the read timeout. The timer is good for the subsequent read request. The time unit is seconds. A zero or negative value indicates that the timeout is disabled. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FCONTROL` (4) and `FDEVICECONTROL` 2 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13115	**Enable/Disable Read Timer(B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.

Table 3-7.
AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13116	**Read Timer (I32) Get: Yes; Put: No; Verify: No; Release 4.0** Returns the amount of time used for completion of the last read in hundreths of a second. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FCONTROL` (22) and `FDEVICECONTROL` 8 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13117	**Line Delete Echo (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the line delete echo status. If this item is set to true, then it will echo `!!!` when the line delete character is used; otherwise, it does not echo the line delete character. Input data is deleted whether or not `!!!` is echoed. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FCONTROL` (34,35) and `FDEVICECONTROL` 14 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13118	**Data Bits (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the data bits per character. When 7 bits are used the current parity setting (see item 13104) controls parity generation and checking. When 8 bit characters are used parity checking is disabled. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 56 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information. Not valid on the physical console.
13119	**Carriage Control Position (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13120	**Enable/Disable Xoff Timer (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13121	**Block Mode Type(I32) Get: Yes; Put: No; Verify: No; Release 4.0** This item returns the type of block mode supported by the driver. Possible return values are: 7 Both line and DTC style block mode 15 PAD terminal supporting page block mode The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 28 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.

Table 3-7.
AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13122	**Enable/Disable Typeahead (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the typeahead enable status. If this item is set to true, then typeahead is enabled. On false, typeahead is disabled. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 51 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13123	**Bypass Typeahead (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the bypass typeahead status. If it is true,the next read should bypass the typeahead buffer and read the data directly from the device. The data in the typeahead buffer is not flushed, and can be obtained by subsequent reads. This function is valid only when typeahead is enabled. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 61 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13124	**Flush Typeahead Data (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This items returns or modifies the flush typeahead status. If this item is true, the driver flushes typeahead buffer. This request is valid for the next read only. This request is only valid if typeahead is enabled. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 60 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13125	**Enable/Disable Console Mode (B) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13126	**Ctl-A read timeout (I32) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13127	**Enable/Disable Device XON/XOFF (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the device XON/XOFF flow control. When device XON/XOFF is enabled, the controller stops sending data to the device when it receives XOFF and resumes when it receives XON. When device XON/XOFF is disabled, the XON and XOFF characters are passed to the host as data. When XON/XOFF flow control is disabled, data overruns can occur. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). This item is not valid on the physical console. See `FDEVICECONTROL` 26 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.

Table 3-7.
AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13128	**XOFF Timer (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the XOFF timer. A positive value, representing a time limit in seconds, enables the timer. A zero or negative value, disables the timer. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). This item is not supported by console device managers. See `FDEVICECONTROL` 27 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13129	**Read Trigger Character (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the read trigger character (normally DC1). A NULL character means there is no read trigger character. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 32 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13130	**Backspace Character (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the back space character in normal editing mode. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 36 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13131	**Line Delete Character (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the line deletion character for normal editing (usually control-X). The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 37 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13132	**End Of Record Character (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the end-of-record character used in edited or unedited mode. A NULL character disables the EOR character. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 39 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13133	**Subsystem Break Character (C) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.

Table 3-7.
AIFDEVICEGET/PUT TERMINAL Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13134	**Enable/Disable Form Feed Character (B) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the directive to allow the substitution of the form feed character in the output stream. When the value is true, the device driver does not substitute the form feed character when it is encountered in the carriage control of terminals. When the value is false, the form feed character is replaced. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). This item is not supported by console device managers. See `FDEVICECONTROL` 52 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13135	**Form Feed Character (C) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the form feed replacement character. Only the form feed in carriage control information will be replaced. The form feed in data is not replaced. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 53 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information. Not supported by console device managers.
13136	**Backspace Response (I32) Get: Yes; Put: Yes; Verify: No; Release 4.0** This item returns or modifies the response action when a backspace character is received. Valid values are: 1 Remove character from input and back cursor up one space 5 Remove character from input and erase character (backspace,space,backspace) The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). See `FDEVICECONTROL` 55 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.
13137	**Last Subsystem Break Character (C) Not Supported** This item is not supported. It was incorrectly documented on Release 4.0 and does not work as previously stated.
13138	**Terminal Type File (Str26) Get:No; Put: Yes; Verify: No; Release 4.0** This item modifies the terminal type or printer type file for use with a device. This file may be created through the `TTUTIL` utility. This is the only item that is supported by serial printers. The behavior of this item varies with the connection type (for example, DTC direct connect, PAD, or Telnet/iX). This item is not supported by console device managers. See `FDEVICECONTROL` 1 in the *Asynchronous Serial Communications Programmer's Reference Manual* for more information.

Printer Device Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with Printer device criteria items used by `AIFDEVICEGET` and `AIFDEVICEPUT`.

Table 3-8. AIFDEVICEGET/PUT PRINTER Device Items from I/O Subsystem

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13201	**Left Margin (I32) Get: No; Put: Yes; Verify: No; Release 4.0** This item allows the caller to get or set the left margin.
13202	**Lines Per Inch (I32) Get: No; Put: Yes; Verify: No; Release 4.0** This item allows the caller to get or set lines per inch. Valid values are either 6 or 8.
13203	**End of Job (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** This item allows the caller to set the end of job.

Tape Device Item Descriptions

The following table provides detailed descriptions of item numbers and correspon ding items associated with TAPE device criteria items used by `AIFDEVICEGET` and `AIFDEVICEPUT`.

Table 3-9. AIFDEVICEGET/PUT TAPE Device Items from I/O Subsystem

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13301	**Fatal Errors (I32) Get: Yes; Put: No; Verify: No; Release 4.0** This item allows the caller to get the fatal error information from the tape generic status. 0 No fatal error 1 Tape runaway 2 Multiple tracks in error 3 Timing error 4 Command reject 5 Unit failure 6 Data parity error 7 Command parity error
13302	**Density (I32) Get: Yes; Put: No; Verify: No; Release 4.0** This item allows the caller to get the density information from the tape generic status. 0 Unrecognized density 1 800 BPI (NZRI) 2 1600 BPI (PE)/ DDS format 3 6250 BPI (GCR)
13303	**Unit Number (I32) Get: Yes; Put: No; Verify: No; Release 4.0** This item allows the caller to get the tape drive unit number. This is not the physical address, but an additional identifier. For HPIB devices and SCSI devices, this will always be zero. When tape devices are supported that use this additional identifier, this field will report that value.
13304	**End of File (B) Get: Yes; Put: No; Verify: No; Release 4.0** Returns true when the tape is positioned at the end of file marker.
13305	**Beginning of Tape (B) Get: Yes; Put: No; Verify: No; Release 4.0** If it returns true, the tape is positioned at the beginning of tape (BOT/ load point)
13306	**End of Tape (B) Get: Yes; Put: No; Verify: No; Release 4.0** If it returns true, the tape is positioned at the end of tape (EOT).
13307	**Immediate Report (B) Get: Yes; Put: No; Verify: No; Release 4.0** Returns true when the tape device is operating in immediate report mode. This means that the device will buffer data until it has enough data to flush to tape. This is the recommended mode of operation.

Table 3-9. AIFDEVICEGET/PUT TAPE Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13308	**Track Error (B) Get: Yes; Put: No; Verify: No; Release 4.0** If it returns true, a single track was found in error.
13309	**Unit Online (B) Get: Yes; Put: No; Verify: No; Release 4.0** If it returns true, the tape drive is online.
13310	**Write Protect (B) Get: Yes; Put: No; Verify: No; Release 4.0** If it returns true, it indicates that the tape is write protected.
13311	**Rewind (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Rewinds the tape to the beginning of tape. NOTE: The tape unit is left online at the end of the rewind.
13312	**Rewind Offline(No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Rewinds tape to the beginning of tape and puts it offline. The tape remains in the drive.
13313	**Write Tape Mark (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Causes one tape mark to be written to tape. Writing tape marks past end-of-tape is permitted.
13314	**Forward File (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Moves the tape forward over the next tape mark but before the next record.
13315	**Backward File (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Moves the tape backward over the previous file mark that is encountered or to the beginning of tape if there is no file mark. The position is immediately in front of the file mark.
13316	**Forward Record (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Moves the tape forward to the beginning of the next record.
13317	**Backward Record (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Moves the tape backward to the beginning of the previous record. If previously positioned at the end of a record, the new position is at the beginning of that record.
13318	**Gap Tape (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Moves the tape forward and erases approximately 3.5 inch of tape. For DDS drives, this does nothing.

Table 3-9. AIFDEVICEGET/PUT TAPE Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13319	**Set Density (I32) Get: No; Put: Yes; Verify: No; Release 4.0** Set the density that a tape will be written at 0 6250 BPI for HP 7978 & 7980 1 1600 BPI for HP 7974, 7978, 7979, all 7980s, and all DDS 2 800 BPI for HP 7974 and some 7978, 7979, and 7980 3 compressed for HP 7980XC and 7980SX 4 no compression for HP 7980XC and 7980SX Density can only be set when a tape is at load point. The density change can vary at the point in which it is displayed. Some devices reflect the change immediately while other devices reflect the density change as the first record is being written. This behavior is device dependent and is not guaranteed to be consistent across all tape drives.
13320	**Set Start/Stop (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Set the HP 7974 to operate in a start/stop mode (50 ipd). This command does nothing on all other devices.
13321	**Set Streaming (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** Set the HP 7974 to operate in a streaming mode. If data is not available to continually write to tape, the drive stops the tape. The drive then repositions itself before it can begin writing again. This item does nothing on all other devices as they are always in this mode.
13322	**Enable/disable Immediate Report (No value) Get: No; Put: Yes; Verify: No; Release 4.0** With immediate report enabled, the device queues up requests. Performance improves when immediate report is enabled.
13325	**Enable/Disable Data Compression (B) Get: No; Put: yes; Verify: no; Release 5.0** Enable or disable data compression on a HPC1504B or HPC1521B DDS drive. This item is not supported on other DDS devices and all 1/2in tapes. The data compression setting will remain in effect until reset via AIFDEVICEPUT or the DEVCTRL.MPEXL.TELESUP program.
13326	**Remote Load (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** This loads a tape to BOT but does not place the drive online. Not valid for the HP 7974 or HP 7978A.
13327	**Remote Unload (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** This unloads a tape and either opens the door (for all 7980s) or ejects the tape (for all DDS). Not valid for the HP 7974 or HP 7978A.

Table 3-9. AIFDEVICEGET/PUT TAPE Device Items from I/O Subsystem (continued)

Item Number	Item Name (Data Type) Get; Put; Verify; Release First Available Description
13328	**Remote Online (No value needed) Get: No; Put: Yes; Verify: No; Release 4.0** This places the drive online. If this item is done to a drive with no tape, or the door is open (7980), the request does not complete until a tape is inserted or the door is closed. Not valid for the HP 7974, HP 7978A/B.
13329	**Enable/Disable Eject (B) Get: No; Put: Yes; Verify: No; Release 5.0** This allows both 1/2in and DDS tape devices to be dynamicaly configured to eject the media following an application rewind/offline or close. This item is not supported on HP 7974, 7978, 7979, 7980, 7980XC, and HP4280 devices. This behavior is device dependent and is not guaranteed to be consistent across all tape drives.

Disk Device Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with Disk device criteria items used by `AIFDEVICEGET` and `AIFDEVICEPUT`.

Table 3-10. AIFDEVICEGET/PUT Disk Device Items from I/O Subsystem

Item Number	Item Name (Data Type) Get; Put; Verify Release First Available Description
13401	**Disk Size (I32) Get: Yes; Put: No; Verify: No; Release 4.0** This returns the disk size in pages (8 sectors per page) from the disk controller.

AIFFILEGGET

Returns global file information.

Syntax

```
                    REC              I32A          @64A
AIFFILEGGET (overall_status, itemnum_array, item_array,
                    RECA             REC     REC      B
             itemstatus_array, UFID, filename, tempfile,
                 I32        REC          REC
             user_id, path_identifier, pathname);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

UFID **record by reference (optional)**

Required if *filename* is omitted. Passes the UFID of the file for which information is desired. Use this parameter if performance is a concern.

Note that this parameter is not adequate for identifying the pathname for an HFS syntax file. You should specify *path_identifier* instead of this item when you are interested in both MPE syntax and HFS syntax files.

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

Default: nil

filename **record by reference (optional)**

Passes the fully qualified name of the file for which information is desired. The name in each element of the record `filename_type` must be left-justified and padded with blanks. In addition, characters must be in the correct case (uppercase and/or lowercase).

If the UFID is omitted and you are interested only in those files that can be represented by MPE syntax, this parameter is required.

For HFS syntax files, the *pathname* parameter should be specified.

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

Default: nil

tempfile **boolean by value (optional)**

Indicates whether or not the file specified is a temporary file. If true, the file is a temporary file. If false, or if this parameter is omitted, the file is a permanent file. (If the file UFID is passed in the *UFID* parameter, this parameter is ignored.)

Default: false

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

path_identifier **record by reference (optional)**

Passes the unique *path_identifier* of the MPE syntax or HFS syntax file for which information is desired. Use this parameter if performance is a concern and you are interested in either an MPE syntax or HFS syntax file. Note that when specifying this parameter, you must specify item 5036 to get the corresponding pathname for an HFS syntax file (for example, /SYS/PUB/dirc/pxfile).

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

Default: nil

pathname **record by reference (optional)**

Passes the pathname of the file (MPE syntax or HFS syntax) for which information is desired. If the *path_identifier* is omitted and you are interested in both MPE syntax and HFS syntax files, this parameter is required. If you specify both this parameter and the *filename* parameter, then the *filename* parameter is ignored. The first word in the pathname specifies the length of the name.

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

Operation Notes

Use *UFID* instead of *filename* for greater performance.

The hierarchical file system (HFS) was incorporated into MPE/iX with release 4.5. The following Operation Notes describe dealing with MPE syntax and HFS syntax files.

MPE Syntax Files

When interested in only those files that can be represented by MPE syntax, the following item keys and items should be used. These items continue to work exactly as they did before the introduction of the hierarchical file system.

Item Keys

- *UFID*
- *filename*

Items

- Item 5002 - *UFID*
- Item 5001 - *file_name*

Note that the UFID key and the UFID item still return valid data for an HFS syntax file since an UFID is still unique for every file on the system; however, the UFID alone is not enough information to identify a unique file name for an HFS syntax file since the file name is no longer kept in the file label.

MPE Syntax and HFS Syntax Files

When interested in all files, the following item keys and items should be used:

Item Keys

- *path_identifier*
- *pathname*

Items

- Item 5037 - *path identifier*
- Item 5036 - *pathname*

Note

Only one item key should be specified. If multiple item keys are specified, then only one key is used and the rest are ignored. The following keys are in order of precedence:

1. *path_identifier*
2. *pathname*
3. *UFID*
4. *filename*

For example, if you specify both the *pathname* and the *UFID* parameters, the *UFID* parameter is ignored. If you specify the *path_identifier* and the *pathname* parameters, then the *pathname* parameter will be ignored.

AIFFILEGPUT

Modifies system global file information.

Syntax

```
                    REC            I32A            @64A
AIFFILEGPUT (overall_status, itemnum_array, item_array,
                    RECA           REC    REC         B       I32
             itemstatus_array, UFID, filename, tempfile, user_id,
                    I32A           @64A        RECA
             ver_item_nums, ver_items, ver_item_statuses,
                    REC            REC
             path_identifier, pathname);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

UFID **record by reference (optional)**

Required if *filename* is omitted. Passes the UFID of the file whose information is to be modified. Use this parameter if performance is a concern.

Note that this parameter is not adequate for identifying the pathname for an HFS syntax file. You should specify *path_identifier* instead of this item when you are interested in both MPE syntax and HFS syntax files.

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

Default: nil

filename **record by reference (optional)**

Passes the fully qualified name of the file for which information is desired. Th e name in each element of the record `filename_type` must be left-justified an d padded with blanks. In addition, characters must be in the correct case (uppercase and/or lowercase).

If the UFID is omitted and you are interested only in those files that can be represented by MPE syntax, this parameter is required.

For HFS syntax files, the *pathname* parameter should be specified.

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

Default: nil

tempfile **Boolean by value (optional)**

Indicates whether or not the file specified is a temporary file. If true, the file is a temporary file. If false, or if this parameter is omitted, the file is a permanent file.

Default: false

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element $n+1$ must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

path_identifier **record by reference (optional)**

Passes the unique *path_identifier* of the MPE syntax or HFS syntax file for which information is desired. Use this parameter if performance is a concern and you are interested in either an MPE syntax or HFS syntax file. Note that when specifying this parameter, you must specify item 5036 to get the corresponding pathname for an HFS syntax file (for example, /SYS/PUB/dirc/pxfile).

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

Default: nil

pathname **record by reference (optional)**

Passes the pathname of the file (MPE syntax or HFS syntax) for which information is desired. If *path_identifier* is omitted and you are interested in both MPE syntax and HFS syntax files, then this parameter is required. If you specify both this parameter and the *filename* parameter, then the *filename* parameter is ignored. The first word in the pathname specifies the length of the name.

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

Operation Notes

If performance is a concern, use *UFID* instead of *filename*.

AIFFILEGGET/PUT Items

The following two tables provide summary and detailed descriptions of the item numbers associated with global file information.

Item Summary

The following table summarizes the item numbers associated with global file information. For more detailed information about these item numbers, refer to the table of global file item descriptions.

Table 3-11. Global File Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
5001	Filename_type	MPE File name	N	Y			
5002	UFID_type	UFID	N	Y			
5003	CA16	Creator name	Y	Y			
5004	Longint_type	Create timestamp	Y	Y			
5005	Longint_type	L A timestamp	Y	Y			
5006	Longint_type	L M timestamp	Y	Y			
5007	Longint_type	F A timestamp	Y	Y			
5008	I32	File code	Y	Y			
5009	U32	File access	Y	Y			
5010	CA8	File lockword	Y	Y			
5011	I32	Unused	N	N			
5012	I32	Foptions	N	Y			
5013	I32	Privileged level	N	Y			
5014	B	Released	Y	Y			
5015	B	Temporary	N	Y			
5016	U32	Record size	N	Y			
5017	U32	End of file (EOF)	N	Y			
5018	U32	File limit	N	Y			
5019	I32	# user labels	N	Y			
5020	I32	User label limit	N	Y			
5021	U32	Block size	N	Y			
5022	I32	Blocking factor	N	Y			
5023	CA34	Volume restriction	N	Y			
5024	I32	Message file open	N	Y			
5025	I32	# of users	N	Y			
5026	I32	# of readers	N	Y			
5027	I32	# of writes	N	Y			
5028	I32	# record pointers	N	Y			
5029	I32	Close disposition	Y	Y	0	5	0
5030	I32	Virtual address	N	Y			
5031	U32	World access	Y	Y			
5032	U32	Group access	Y	Y			
5033	U32	Group librarian access	Y	Y			
5034	U32	Account access	Y	Y			
5035	U32	Account librarian access	Y	Y			

Table 3-11. Global File Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
5036	pathname_type	Pathname	N	Y			
5037	path_identifier	Path Identifier	N	Y			
5038	U32	Running Link Count	N	Y			
5039	U32	File type	N	Y			
5040	U32	Record type	N	Y			
5041	CA36	File Owner	Y	Y			
5042	B	ACD Required	N	Y			
5043	CA16	File Group	N	Y			
5044	logint_type	State Change Timestamp	Y	Y			
5045	B	Update State Change Timestamp	Y	N			
5046	U32	Current Link Count	N	Y			
5047	I32	Number of Extents	N	N			
5048	I32	Number of Sectors	N	N			
5051	B	Follow Symbolic Link	N	N			

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with global file information.

Table 3-12. Global File Information Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5001	**MPE File Name (REC) Put: No; Verify: Yes; Release 3.0** Returns the fully qualified file name in the record format defined by `filename_type`. The file name, group name, and account name are each left-justified and padded with blanks. Note that this item should only be used for names that can be expressed using MPE-only semantics (for example, NL.PUB.SYS). Item 5036 should be used for HFS syntax or MPE syntax files that will be represented using an HFS pathname (for example, /SYS/PUB/pxdir/pxfile). If you specify this item for an HFS syntax file, blanks and a warning are returned in `itemstatus_array`. Record type: `filename_type` (Refer to appendix B.)
5002	**UFID (REC) Put: No; Verify: Yes; Release 3.0** Returns the UFID associated with a file name in the record format defined by *ufid_type*. Record type: `ufid_type` (Refer to appendix B.)
5003	**Creator Name (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the name of the user who created the file. It is assumed that the account name of the file and the creator are the same. This should be a legal MPE/iX user name that is left-justified and padded with blanks. Note that with the introduction of the hierarchical file system, the creator concept has been replaced with the concept of the file owner. The creator name item is maintained for backward compatibility, but in the future, you should use item 5041 to obtain or modify the full file owner name (user and account).
5004	**Creation Timestamp (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of microseconds from January 1, 1970 to the time that the file was created. Record type: `longint_type` (Refer to appendix B.)
5005	**Last Access Timestamp (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of microseconds from January 1, 1970 to the time that the file was last accessed. Record type: `longint_type` (Refer to appendix B.)
5006	**Last Modify Timestamp (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of microseconds from January 1, 1970 to the time that the file was last modified. Record type: `longint_type` (Refer to appendix B.)

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5007	**File Allocation Timestamp (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of microseconds from January 1, 1970 to the time that the file was allocated. Record type: `longint_type` (Refer to appendix B.)
5008	**File Code (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the file code of the file. A negative number indicates that the file is privileged.
5009	**Creator Access Rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the file creator access rights. Bits and their meanings are: Bits (0:24) Unused Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read
5010	**File Lockword (CA8) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the lockword of the file. This should be a legal MPE/iX lockword, left-justified and padded with blanks. If a file does not have a lockword, blanks are returned. You cannot use this item for files outside MPE groups since files in HFS directories cannot have lockwords. POSIX does not recognize the concept of lockwords, and they cannot be specified using POSIX syntax.
5012	**Foptions (I32) Put: No; Verify: Yes; Release 3.0** Returns the file characteristics in the form of an foptions bit mask. The bits and their meanings are Bits (0:18) Unused Bits (18:3) File type Bit (21:1) File equations Bit (22:1) Labeled tape Bit (23:1) Carriage control Bits (24:2) Record type Bits (26:3) File designator Bit (29:1) ASCII/binary Bits (30:2) File domain
5013	**Privileged Level (I32) Put: No; Verify: Yes; Release 3.0** Returns the privileged level of the file. This should be a value from 0 to 3 where the lower the number, the higher the privileged level.

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5014	**Released (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies whether the file is released or secure. True when the file is released and false when the file is secure. This file aspect can be changed using the `RELEASE` and `SECURE` commands if you are the file's creator.
5015	**Temporary (B) Put: No; Verify: Yes; Release 3.0** Returns true if the file is temporary and false if it is permanent.
5016	**Record Size (U32) Put: No; Verify: Yes; Release 3.0** The file's record size in bytes. For variable-length records, the size of the largest record is returned.
5017	**End of File (U32) Put: No; Verify: Yes; Release 3.0** Returns the current number of bytes in the file. This item should be used as an unsigned integer. The number of records in the file can be calculated by the formula: `#of records =(EOF - (256*Number of User Labels))/Record Size`
5018	**File limit (U32) Put: No; Verify: Yes; Release 3.0** Returns the maximum number of bytes that the file is allowed to have. This item should be used as an unsigned integer.
5019	**Number of User Labels (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of user labels that have been allocated for this file. This should be a value from 0 to 254.
5020	**User Labels Limit (I32) Put: No; Verify: Yes; Release 3.0** Returns the end of the user label area as a byte offset. This value can be divided by $100 to calculate the number of user labels written.
5021	**Block Size (U32) Put: No; Verify: Yes** Returns the block size of the file in bytes. This should be the record size multiplied by the blocking factor.
5022	**Blocking Factor (I32) Put: No; Verify: Yes; Release 3.0** Returns the blocking factor for the file. This is the number of records that will be placed in each block and should be a value from 1 to 255.
5023	**Volume Restriction (CA34) Put: No; Verify: Yes; Release 3.0** Returns the file's volume restrictions. The last two characters indicate what the first 32 characters represent, as follows: ' 0' File restricted to the volume name located in elements 1..32. ' 1' File restricted to the volume class name located in elements 1..32. ' 2' File restricted to the volume set name located in elements 1..32.

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5024	**Message File Open/Close Record Count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of open/close records. This is valid for message files only. For non-message files this value is zero.
5025	**Number of Users (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of users that have this file open on the system.
5026	**Number of Readers (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of users that have this file open with read access. This is valid only for NM files. (Not applicable to the system libraries.)
5027	**Number of Writers (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of users that have this file open with write access. This is valid only for NM files.
5028	**Number of Record Pointers (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of record pointers that are active for this file. This is valid only for NM files.
5029	**Close Disposition (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the disposition to be used when the file is closed by the last user accessing it. Values and their meanings are: 0 Null 1 Permanent 2 Temporary (rewind upon close) 3 Temporary (do not rewind) 4 Purge 5 Permanent to temporary (native mode files only) Equivalent to the `HPFOPEN` final disposition option.
5030	**Virtual Address (@64) Put: No; Verify: Yes; Release 3.0** Returns the virtual address of the file.
5031	**Any Access Rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the access rights for any user. Bits (0:24) Unused Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5032	**Group Access Rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the group access rights. Bits (0:24) Unused Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read
5033	**Group Librarian Access Rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the group librarian access rights. Bits (0:24) Unused Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read
5034	**Account Access Rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the account access rights. Bits (0:24) Unused Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5035	**Account Librarian Access Rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the account librarian access rights. Bits (0:24) Unused Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read
5036	**Pathname (REC) Put: No; Verify Yes; Release 4.5** Returns the pathname in the record format defined by `pathname_type`. The name is returned as an absolute pathname (for example, /SYS/PUB/NL). Record type: `pathname_type`
5037	**Path Identifier (REC) Put: No; Verify: Yes; Release 4.5** Returns the unique path identifier associated with the specified pathname. Record type: `path_identifier`
5038	**Running Link Count (U32) Put: No; Verify: Yes; Release 4.5** Returns the total number of links that have been performed on this file since it was created. It does not indicate the current number of links; it is simply a running count. Use item 5046 to get the current number of links.
5039	**File type (U32) Put: No; Verify: Yes; Release 4.5** Returns the file type. Possible values are: 0 - ordinary 1 - ksam 2 - relative_io 3 - nm_ksam 4 - circular 5 - spool 6 - message 7 - resv 8 - cmfile 9 - dir_obj 10 - label_table 11 - xm_syslog 12 - pipe 13 - fifo 14 - symbolic link 15 - device link

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5040	**Record type (U32) Put: No; Verify: Yes; Release 4.5** Returns the file record type. Possible values are: 0 - fixed 1 - variable 2 - undefined 3 - cm_spool 4 - account directory node 5 - user directory node 6 - group directory node 7 - fileset directory node 8 - temporary directory 9 - byte_stream 10 - hierarchical directory
5041	**File Owner (CA36) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the full file owner name. The name is in the format USER.ACCOUNT and is padded with blanks. Note that this item should be used instead of item 5003 to get the full file owner. With the Hierarchical File System, the creator field has been replaced with the concept of a file owner. This is because file ownership can now change through the use of the `chown` function. Note that for MPE directory files existing prior to MPE/iX release 4.5, the creator field will not be initialized. For those files, blanks will be returned.
5042	**ACD Required (B) Put: No; Verify: Yes; Release 4.5** Returns true if the file must have an ACD to establish its security policy. If this flag is false, the file may have an ACD in effect.
5043	**File_group (CA16) Put: No; Verify: Yes; Release 4.5** Returns the name of the file sharing group that the file belongs to. This field is the text form of the 32 bit GID value from the HPGID.PUB.SYS file. This value is inherited from the parent directory and may change through the use of the `HPSETOWNER` intrinsic.
5044	**State Change Timestamp (REC) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the number of microseconds from January 1, 1970 to the time that the file label was last changed. This timestamp is a new field in the file label which was added to meet POSIX standards. Although this item gives you control over the timestamp, it should be updated whenever the file label changes. If you do not specify this item or item 5045, then the timestamp is updated automatically with the current time during any `AIFFILEGPUT` operation that affects the file label. Record type: `longint_type` (Refer to Data Type Definition.)
5045	**Update State Change Timestamp? (B) Get: No; Put: Yes; Verify: No; Release 4.5** Indicates whether or not to automatically update the state change timestamp during any `AIFFILEGPUT` operation which effects the file label. The current time will be used unless a value is specified in item 5044. If item 5044 is specified, then the timestamp will be updated with the user specified value regardless of whether or not this flag is specified. Defaults to true (that is, update timestamp automatically).

Table 3-12. Global File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release Available Description
5046	**Current Link Count (U32) Put: No; Verify: Yes; Release 4.5** Returns the current number of links (hard links) for this file.
5047	**Number of extents (I32) Put: No; Verify: Yes; Release 4.5** Number of extents used by the file.
5048	**Number of Sectors (I32) Put: No; Verify: Yes; Release 4.5** Number of sectors used by a file.
5051	**Don't Follow Symbolic Link (B) Put: No; Verify: Yes; Release 5.0** This item is used as an option to determine whether the information returned by `AIFFILEGGET` is about the symbolic link or the resolved link (resolution of the symbolic link). The default is to resolve the symbolic link. This is only valid when either a pathname or MPE file name is passed to `AIFFILEGGET`. For pathnames, true means `AIFFILEGGET` does not resolve the last component of a pathname if it is a symbolic link. For pathnames, false means the full path is resolved. To locate symbolic links, use 14 the symbolic link file type (of item 5039) with `AIFSYSWIDEGET` (The callers can control the information returned by either setting the passed value to true or false).

AIFFILELGET

Returns process-specific file information.

Syntax

```
                    REC             I32A           @64A
AIFFILELGET (overall_status, itemnum_array, item_array,
                     RECA          I32    REC    REC      I32
             itemstatus_array, fnum, PID, UFID, user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

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

Passes the process-specific file number for which information is desired.

PID **record by value (optional)**

Passes the *PID* of the process for which information is desired. The default is the current process.

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

Default: 0

UFID **record by reference (optional)**

Passes the unique file identifier of the MPE or HFS file about which information is desired. If you are using the path identifier item key from `AIFSYSWIDEGET` or from `AIFPROCGET`, then you need to specify the path_identifier.ufid field for this parameter. The default is no UFID checking.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

The ***fnum*** parameter passes the file number returned by the file system to the calling process at open (`FOPEN/HPFOPEN`) time. It is the number used to invoke the various file system intrinsics. *PID* passes the PID of the process that issued the `FOPEN/HPFOPEN` call. *PID* is optional and defaults to the calling process's PID. If there is no active process associated with *PID*, `AIFFILELGET` returns an error condition. If there is no process-specific active file associated with ***fnum***, `AIFFILELGET` returns an error condition.

The PID/file number pairs are obtainable in the following ways:

- If no *PID* is passed, use the file numbers passed by the file system at open time.
- Use `AIFPROCGET`, specifying a PID or PIN and item number 2063 `File numbers of open files`.
- Use `AIFFILELGET`, specifying the item `List of sharers`.

Since the PID/file number pair does not specify a file unique over the lifetime of a process, there is provision for accepting a UFID as a confirmation key. The PID/file number pair selects a particular file on the system, which can then be confirmed uniquely by matching its UFID with the UFID passed. If the confirmation fails, the `AIFFILEGGET` returns an error condition. If no UFID is passed, no such check is carried out.

`AIFFILEGGET` is designed to make the differences between NM and CM files transparent. Thus, it can determine whether the file is an NM file or not. If it is an NM file, only the NM structures are accessed. However, if it is a CM file, the CM structures (PACB, LACB) need to be accessed.

Note that this AIF will return an error (-33, "Invalid Fnum PID combination"), if an attempt is made to retrieve information for a remote file.

Operation Notes - HFS

MPE Files

When you are interested in MPE file names only, the following items should be used. These items will continue to work exactly as they did before the introduction of the Hierarchical File System.

ITEMS

- Item 4001 - *filename*
- Item 4002 - *UFID*

Note that the *UFID* item will still return valid data for an HFS file since a *UFID* is still unique for every file on the system. However, the *UFID* alone will not be enough information to identify a unique filename for a HFS file since the filename is no longer kept in the file label.

MPE and HFS Files

When interested in all files, the following items should be used:

ITEMS

Item 4036 - *pathname*

Item 4037 - *path_identifier*

Items Returned for Directory Files

Prior to POSIX, these AIFs would return the value 0 for certain items when the file specified was a DIRECTORY file. Since users can now open DIRECTORY files, values other than 0 may be returned for these items. Below is a list of the items which would previously return 0 for DIRECTORY files:

```
Item 4010 - Record pointer
Item 4011 - Record number
Item 4012 - Offset within block
Item 4014 - Multiaccess type
Item 4015 - # of MULTI sharers
Item 4016 - MULTI sharer lock
Item 4017 - PIDs and file number of sharers
Item 4022 - # of records transferred
Item 4033 - File pointer offset
Item 4034 - # bytes read
Item 4035 - # bytes written
```

You should exercise caution when retrieving the list of file sharers (item 4017) for the system directory file, $ROOT, since every process will have this file opened. System performance could be adversely effected.

AIFFILELPUT

Modifies process-specific file information.

Syntax

```
                    REC              I32A            @64A
AIFFILELPUT (overall_status, itemnum_array, item_array,
                     RECA          I32     REC    REC     I32
             itemstatus_array, fnum, PID, UFID, user_id,
                    I32A          @64A           RECA
             ver_item_nums, ver_items, ver_item_statuses);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

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

Passes the process-specific file number of a file whose information is to be modified.

PID **record by value (optional)**

Passes the PID of the process associated with a file whose information is to be modified. The default is the current process.

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

Default: 0

UFID **record by reference (optional)**

Passes the unique file identifier of an MPE or HFS file whose information is to be modified. If you are using the Path Identifier item key from `AIFSYSWIDEGET` or from `AIFPROCGET`, you will need to specify the path_identifier.ufid field for this parameter.

The default is no UFID checking.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element $n+1$ must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

Operation Notes

The ***fnum*** parameter passes the file number returned by the file system to the calling process at open (`FOPEN/HPFOPEN`) time. It is the number used to invoke the various file system intrinsics. *PID* passes the PID of the process that issued the `FOPEN/HPFOPEN` call. *PID* is optional and defaults to the calling process's PID. If there is no active process associated with *PID*, `AIFFILELGET` returns an error condition. If there is no process-specific active file associated with ***fnum***, `AIFFILELGET` returns an error condition.

The PID/file number pairs are obtainable in the following ways:

- If no *PID* is passed, use the file numbers passed by the file system at open time.
- Use `AIFPROCGET`, specifying a PID or PIN and item number 2063 `File numbers of open files`.
- Use `AIFFILELGET`, specifying the item `List of sharers`.
- Use `AIFSYSWIDEGET`, specifying the item `File opened`.

Since the PID/file number pair does not specify a file unique over the lifetime of a process, there is provision for accepting a UFID as a confirmation key. The PID/file number pair selects a particular file on the system which can then be confirmed uniquely by matching its UFID with the UFID passed. If the confirmation fails, the `AIFFILELPUT` returns an error condition. If no UFID is passed, no such check is carried out.

`AIFFILELPUT` is designed to make the differences between NM and CM files transparent. Thus, it can determine whether the file is an NM file or not. If it is an NM file, only the NM structures are accessed. However, if it is a CM file, the CM structures (PACB, LACB) must be accessed.

The processes whose files are accessible through `AIFFILELPUT` are user processes. Processes of type `SYSTEM`, `UCOP`, `MAIN`, and `DETACH` are not accessible. If one of these processes is specified, `AIFFILELPUT` terminates with an error condition.

This procedure can be used to modify the attributes of only the user files. The files excluded are those with local file numbers zero through eight and those with file designation other than user. If you attempt to modify the information about one of the excluded files, `AIFFILELPUT` terminates with an error condition.

Note that this AIF will return an error (-33, "Invalid Fnum PID combination"), if an attempt is made to retrieve information for a remote file.

AIFFILELGET/PUT Items

The following two tables provide summary and detailed descriptions of the item numbers associated with local (process-specific) files.

Item Summary

The following table summarizes the item numbers associated with local (process-specific) file information. For more detailed information about these item numbers, refer to the table of local file item descriptions.

Table 3-13. Local File Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
4001	Filename_type	File name	N	Y			
4002	UFID_type	UFID	N	Y			
4003	I32	File number	N	Y			
4004	I32	File designation	N	Y			
4005	B	NOWAIT IO?	N	Y			
4006	B	Buffered access?	N	Y			
4007	B	Multiple record I/O?	N	Y			
4008	B	Short mapped?	N	Y			
4009	I32	Short mapped count	N	Y			
4010	@64	Record pointer	N	Y			
4011	I32	Record number	Y	Y	0	-1	0
4012	I32	Offset within block	Y	Y	0	-1	-4016
4013	I32	Open count	N	Y			
4014	I32	MULTIaccess type	N	Y			
4015	I32	# of MULTI sharers	N	Y			
4016	I32	MULTI sharer lock	Y	Y	0	2	-4012
4017	RecFNumPID_type	Sharer PIDs/fnums	N	N			
4018	longint_type	# logical reads	Y	Y			
4019	longint_type	# logical writes	Y	Y			
4020	U32	# records read	Y	Y			
4021	U32	# records written	Y	Y			
4022	Longint_type	# records transferred	Y	Y			
4023	I32	Bytes transferred last I/O	Y	Y			
4024	B	CM file?	N	Y			
4025	I32	Last error	Y	Y			
4026	U32	Access rights	Y	Y	0	255	-4010
4027	I32	Input priv level	Y	Y			
4028	I32	Output priv level	Y	Y	2	3	-4011
4029	I32	Access priv level	Y	Y	2	3	-4011
4030	B	I/O outstanding?	N	Y			
4031	B	Device file ?	N	Y			
4032	B	Directory object?	N	Y			

Table 3-13. Local File Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
4033	U32	File pointer offset	Y	Y	0	-1	-4009
4034	Longint_type	# bytes read	Y	Y			
4035	Longint_type	# bytes written	Y	Y			
4036	pathname_type	Pathname	N	Y			
4037	path_identifier	Path Identifier	N	Y			
4038	B	Opened by UFID	N	Y			
4039	B	Close on Exec	N	Y			
4040	B	Append Mode	N	Y			
4041	B	Non-Block Mode	N	Y			

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with local (process-specific) file information.

Table 3-14. Local File Information Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4001	**MPE File Name (REC) Put: No; Verify: Yes; Release 3.0** Returns the fully qualified file name of the MPE file. The file name, group name, and account name are each left-justified and padded with blanks. Note that this item should only be used for names that can be expressed using MPE-semantics (for example, NL.PUB.SYS). Item 4036 should be used for HFS syntax or MPE syntax files which are represented using a HFS pathname (for example, /SYS/PUB/pxdir/pxfile). If you select this item for a file that cannot be expressed using MPE-only semantics, then blanks are returned and a warning is returned in itemstatus_array. Record type: `filename_type` (Refer to appendix B.)
4002	**File UFID (REC) Put: No; Verify: Yes; Release 3.0** Returns the UFID of the file. The UFID is unique for all files on the system. Note that for HFS files, you should be selecting the path_identifier (item 4037). Although every file has a unique UFID, the linkid and parent_ufid are needed to quickly identify a unique pathname for HFS files, since POSIX will introduce the concept of multiple file links/aliases in the future. Record type: `ufid_type` (Refer to appendix B.)
4003	**File number (I32) Put: No; Verify: Yes; Release 3.0** Returns the process-specific file number assigned to the file at every `HPFOPEN/FOPEN` issued by the process.
4004	**File designation (I32) Put: No; Verify: Yes; Release 3.0** Returns the designation assigned to the file at `HPFOPEN/FOPEN` time. Each process has some standard file numbers assigned and opened by the system on behalf of the user. Any user-issued calls are assigned the designation 'user'. Values and their meanings are: 0 User file 1 `$STDLIST` 2 `$NEWPASS` 3 `$OLDPASS` 4 `$STDIN` 5 `$STDINX` 6 `$NULL`
4005	**NOWAIT I/O? (B) Put: No; Verify: Yes; Release 3.0** Returns the file's NOWAIT I/O status. True indicates that the process blocks for I/O (NOWAIT I/O) against the specified file. False indicates that NOWAIT I/O is not set. NOWAIT I/O is set at open time. For `FOPEN`, it corresponds to setting *aoptions* (4:1) and for `HPFOPEN`, it corresponds to the specification of item 16.

Table 3-14. Local File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4006	**Buffered access? (B) Put: No; Verify: Yes; Release 3.0** Returns the file's buffering status. True indicates that the file system uses buffering to access the specified file. False indicates no buffered access. Buffered access is set at open time. For `FOPEN`, it corresponds to setting *aoptions* (7:1) and for `HPFOPEN`, it corresponds to the specification of item 46.
4007	**Multiple record I/O? (B) Put: No; Verify: Yes; Release 3.0** Returns the file's multiple record I/O status. True indicates that the file system transfers multiple records in a single read or write operation against the specified file. False indicates no multiple record transfer. For `FOPEN`, it corresponds to setting *aoptions* (11:1) and for `HPFOPEN` it corresponds to the specification of item 15.
4008	**Is file short mapped? (B) Put: No; Verify: Yes; Release 3.0** Returns whether or not the file is short mapped. True indicates that the file is short mapped and false otherwise. Short-mapped access is specified through `HPFOPEN`, item 18.
4009	**Short mapped count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of times this file is currently opened in short-mapped mode by the specified process.
4010	**Record pointer (@64) Put: No; Verify: Yes; Release 3.0** Returns the virtual address pointed to by the file's record pointer. It is the address of the next byte that will be read or written. If the file is being shared MULTI, this pointer points to the next byte for I/O for all the sharers. Valid only for NM files.
4011	**Record number (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of the record pointed to by the specified file's record pointer. If the file is being accessed MULTI, then this is the number of the record pointed by the group of MULTI sharers of which this file is a member. If you are modifying the record number, the number you pass must not exceed the number of records in the file. If it does, the next I/O for this file will lead to a system abort. In addition, be sure to modify both the record number and the offset in a consistent manner.
4012	**Offset within current block (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the offset, in bytes, within the record block indicated by the record pointer. If the file is being accessed MULTI, then this is the offset pointed to by the group of MULTI sharers of which this file is a member. Valid only for variable-length record files. Be sure to modify both the record pointer and the record number in a consistent manner.
4013	**Open count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of outstanding opens against the specified file by the specified process.

Table 3-14. Local File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4014	**Multiaccess type (I32) Put: No; Verify: Yes; Release 3.0** Returns the type of multiaccess specified for the specified file at open time, indicating how the record pointer is to be shared. For `FOPEN` it corresponds to *aoptions* (5:2) and for `HPFOPEN`, to item 14. Valid values and their meaning are as follows: 0 No multi 1 Intrajob 2 Interjob
4015	**Number of MULTI sharers (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of opens sharing the record pointer of this file (includes the open indicated by the specified file number).
4016	**Locking for MULTI sharers (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the type of lock placed on the specified file (valid for the group of MULTI sharers to which the file number belongs). This lock is placed at open time and operates for subsequent attempts to open this file and share the record pointer. It does not reflect the lock operating for attempts to open the file without sharing the record pointer. (Modifying this information has effect only upon subsequent opens trying to share the record pointer.) For `FOPEN` it corresponds to aoptions (8:2) and for `HPFOPEN` to `HOP_OPTION_EXCLUSIVE`. Valid values and their meanings are as follows: 0 Default 1 Exclusive 2 Exclusive Access Read 3 Share

4017	**PIDs and file numbers of sharers (REC) Put: No; Verify: No; Release 3.0**
	
	Returns an array of records with the following Pascal declaration:
	
	`Record`
	`  fnum : integer;`
	`  PID  : longint;`
	`End;`
	
	Each element contains the PID of the process that has an open sharing the record pointer, and the file number of the open that shares the pointer. If a process has more than one file number sharing the record pointer, its PID appears twice. Valid only for NM files.
	
	You should pass an area of appropriate size. The first word of the buffer is expected to hold the size, in 3-word units, of the rest of the buffer area. The first word, upon return, specifies the number of records returned. Check the appropriate *itemstatus_array* element to determine whether or not information was truncated (because the area you passed was not of sufficient size to hold all of the information).

4018	**Number of logical reads (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of logical reads made against the specified file. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid only for NM files. Record type: `longint_type` (Refer to appendix B.)

Table 3-14. Local File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4019	**Number of logical writes (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of logical writes made against the specified file. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid only for NM files. Record type: `longint_type` (Refer to appendix B.)
4020	**Number of records read (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of records read from the specified file. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid only for NM files.
4021	**Number of records written (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of records written to this file number. Valid only for NM files. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics.
4022	**Number of records transferred (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of records transferred to and from the specified file. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid for NM and CM files. Record type: `longint_type` (Refer to appendix B.)
4023	**Bytes transferred in last I/O (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of bytes transferred (input or output) to or from the specified file in the last I/O operation. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid only for NM files.
4024	**CM file? (B) Put: No; Verify: Yes; Release 3.0** Returns true if the file is a CM File. This information is useful in determining whether or not a file is NM in order to use particular AIF items appropriate only to NM files or to CM files.
4025	**Last error (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the last file system error for the specified file, interpreted as `status_type` (refer to appendix B). Valid only for NM files.

Table 3-14. Local File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4026	**Access rights (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies a bit mask indicating the access rights for the specified file. File access rights dictate the kind of operations permitted to the process. If a bit is set to 1, the process has that right. It is specified at open time. For `FOPEN`, it corresponds to *aoptions* (12:4) and (10:1). For `HPFOPEN`, it corresponds to items 11 and 12. Bits and their corresponding access rights are as follows: Bits (0:24) Unused (set to zero) Bit (24:1) Read Bit (25:1) Write Bit (26:1) Execute Bit (27:1) Append Bit (28:1) Lock Bit (29:1) Save Bit (30:1) Update Bit (31:1) Dir_read
4027	**Input privileged level (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum privileged level for the specified process to read from the specified file. This privileged level is applicable to all file numbers of the process corresponding to the specified file. It is set at open time. Input privileged level also depends upon the privileged level of the user and the specified access rights. Valid only for NM files. For `FOPEN` the access rights are specified through *aoptions* (12:4) and for `HPFOPEN` they are specified through item 11. It is used only for mapped reads from the file. If there are multiple opens of the same file, this is the least restrictive of all of the individual opens. Modifying this information has effect only upon the succeeding attempts to read the mapped pages for the file. Only files set to input privileged levels 2 and 3 can be accessed. Valid values are 2 or 3.
4028	**Output privileged level (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum privileged level for the specified process to write to the specified file. This output privileged level is applicable to all file numbers of the process corresponding to the specified file. It is set at open time. Output privileged level also depends upon the privileged level of the user and the specified access rights. Valid only for NM files. For `FOPEN` the access rights are specified through *aoptions* (12:4) and for `HPFOPEN` they are specified through item 11. It is used only for mapped writes to the file. If there are multiple opens of the same file, this is the least restrictive of all of the individual opens. Modifying this information has effect only upon the succeeding attempts to write to the mapped pages for the file. Only files set to output privileged levels 2 and 3 can be accessed. Valid values are 2 or 3.

Table 3-14. Local File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4029	**Access privileged level (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum privileged level for accessing the specified file. This is set at open time. For `FOPEN`, it defaults to the user's privileged level. For `HPFOPEN`, it corresponds to item 29. Access privileged level determines the file system intrinsics callable for the specified file and the privileged level required for access. Modifying this information has effect only upon the succeeding attempts to access the file through file system intrinsics. Only files set to output privileged levels 2 and 3 can be accessed. Valid values are 2 or 3.
4030	**I/O outstanding? (B) Put: No; Verify: Yes; Release 3.0** Returns true if there is I/O outstanding for the specified file. The system sets this information to true when the process issues an `FREAD` or an `FWRITE` whether or not NOWAIT I/O was specified at open time. It is then set to false until the file system call returns to the caller in the waited I/O case. For files opened NOWAIT I/O, this information remains true until the user issues a successful call to `IOWAIT` or `IODONTWAIT`. In any case, a call to `FCONTROL`, specifying controlcode 43, causes this information to be set to false.
4031	**Device file? (B) Put: No; Verify: Yes; Release 3.0** Returns true if the specified file is a device file (false otherwise). This information determines whether or not some item information can be returned or modified.
4032	**Directory object? (B) Put: No; Verify: No; Release 3.0** Returns true if the specified file is a directory object (for example, group node, account node, fileset node, hierarchical directory). This includes directory object files opened on behalf of the user by the system and those directory files explicitly opened by the user.
4033	**Offset to file pointer (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the offset to the file pointer, from the beginning of the file. When modifying this information, you must ensure that the new offset does not point outside the current file limits. If this occurs, the next I/O call leads to a system abort. In addition, it is your responsibility to update the record number and the offset within it to be consistent with the new record pointer. Failing to do so leads to unpredictable behavior. In any case, the normal protection of virtual memory is enforced during I/O.
4034	**Bytes read (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of bytes read. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid only for NM files. Record type: `longint_type` (Refer to appendix B.)

Table 3-14. Local File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
4035	**Bytes written (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of bytes written. This information is kept for accounting and measurement interface purposes. Modifying this information affects only the concerned statistics. Valid only for NM files. Note: This item reflects the actual number of bytes written to disk. This item will contain a value > 0 for a NM KSAM file even if the file is only read from, because information in the KSAM control block is updated and written to disk even on a FREAD (such as the counters which keep track of the number of FREADs). Record type: `longint_type` (Refer to appendix B.)
4036	**Pathname (REC) Put: No; Verify: Yes; Release 4.5** Returns the absolute pathname of the file. If the user opened the file by UFID and not by name, then this item will return blanks and a warning. On input, the first four bytes in the buffer will represent the maximum buffer length in bytes. On output, the name will be terminated by a NULL character and the first four bytes will contain the actual number of bytes returned (not including the NULL character or the one word length). Record type: `pathname_type` (Refer to appendix B.)
4037	**Path Identifier (REC) Put: No; Verify: Yes; Release 4.5** Returns the unique path identifier of the file. If the file was opened by UFID and not by name, then the parent_ufid and link ID will be 0 and a Warning will be returned. In this case, the path identifier will not be sufficient to return a pathname for an HFS file. Record type: `path_identifier` (Refer to appendix B.)
4038	**Opened by UFID (B) Put: No; Verify: Yes; Release 4.5** Returns true if the file was opened by UFID and not by name. If the file was opened by UFID, then the pathname, the parent_ufid, and the link ID will not be known. See the descriptions for the pathname item, 4036, and the path identifier item, 4037, for more information.
4039	**Close on Exec (B) Put: No; Verify: Yes; Release 5.0** Returns whether the close on exec flag has been set for the file. If the flag is set, then the file will be closed upon successful execution of the exec family functions.
4040	**Append Mode (B) Put: No; Verify: Yes; Release 5.0** Returns whether the file is in append mode. If this flag is set, the file offset will be set to the end of the file prior to each write.
4041	**Non-Block Mode (B) Put: No; Verify: Yes; Release 5.0** Returns whether the file is in non-block mode. This flag is relevant to character special files such as fifos, pipes, etc.

AIFGLOBACQ

Allocates an object of a specified size and places a pointer to that object in the Architected Interface Facility: Operating System internal data area.

Syntax

```
                    REC          I32   I32     REC
AIFGLOBACQ (overall_status, user_id, size, user_cell);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

user_id **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: Operating System product.

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

Passes the size, in bytes, of the object to be allocated. The maximum size is 2**32 bytes long.

user_cell **record by reference (required)**

Returns a pointer to an object of the specified size. This pointer is also placed in the Architected Interface Facility: Operating System internal data area.

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

Operation Notes

The acquired object can be released by AIFGLOBREL The acquired object does not survive system reboots.

The *user_cell* parameter can also contain an address from the previous call, a bit map, or a customer 64-bit representation. It is the application designer's responsibility to save and manage this field.

AIFGLOBACQ, AIFGLOBREL, AIFGLOBGET, and AIFGLOBPUT all access the user cell. The user cell is a 64-bit data area which is located in the AIF internal area. All processes with the same user_id will share the same user_cell. The management of the user cell is the application designer's responsibility.

When AIFGLOBACQ is called, it returns the address of the newly allocated object in the user_cell. If any value is in the user_cell prior to calling AIFGLOBACQ it is the user's responsibility to save the value. Likewise, if the user_cell is modified using AIFGLOBPUT, the application designer must save the value returned by AIFGLOBACQ. The value returned in the user_cell when the object was first allocated using AIFGLOBACQ must be placed back in the user_cell with AIFGLOBPUT if the user_cell has been modified between calls to AIFGLOBACQ and AIFGLOBREL.

AIFGLOBGET

Returns the contents of the user cell in the Architected Interface Facility: Operating System internal data area.

Syntax

```
                  REC          I32       REC
AIFGLOBGET (overall_status, user_id, user_cell);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

user_id **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: Operating System product.

user_cell **record by reference (required)**

Returns the contents of the user cell in the internal data area.

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

Operation Notes

This user cell is shareable among all processes using the same user ID.

`AIFGLOBACQ`, `AIFGLOBREL`, `AIFGLOBGET`, and `AIFGLOBPUT` all access the user cell. The user cell is a 64-bit data area which is located in the AIF internal area. All processes with the same user_id will share the same user_cell. The management of the user cell is the application designer's responsibility.

When `AIFGLOBACQ` is called, it returns the address of the newly allocated object in the user_cell. If any value is in the user_cell prior to calling `AIFGLOBACQ` it is the user's responsibility to save the value. Likewise, if the user_cell is modified using `AIFGLOBPUT`, the application designer must save the value returned by `AIFGLOBACQ`. The value returned in the user_cell when the object was first allocated using `AIFGLOBACQ` must be placed back in the user_cell with `AIFGLOBPUT` if the user_cell has been modified between calls to `AIFGLOBACQ` and `AIFGLOBREL`.

AIFGLOBINSTALL

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

Syntax

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

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

user_id **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: Operating System product.

Operation Notes

`AIFGLOBINSTALL` is the programmatic equivalent of executing the `INSTOS` installation utility. `AIFGLOBINSTALL` (or `INSTOS`) must be executed on all systems containing code that calls operating system 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 operating system 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`.

AIFGLOBLOCK

Restrict access to the user cell located in the Architected Interface Facility: Operating System internal data area.

Syntax

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

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

user_id **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: Operating System product.

Operation Notes

None.

AIFGLOBPUT

Places a user-defined value (for example, a pointer or a bit map) in the user cell of the Architected Interface Facility: Operating System internal data area.

Syntax

```
                  REC          I32       REC
AIFGLOBPUT (overall_status, user_id, user_cell);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

user_id **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: Operating System product.

user_cell **record by value (required)**

Passes a user-defined value to be placed in the user cell of the Architected Interface Facility: Operating System internal data area.

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

Operation Notes

`AIFGLOBACQ`, `AIFGLOBREL`, `AIFGLOBGET`, and `AIFGLOBPUT` all access the user cell. The user cell is a 64-bit data area which is located in the AIF internal area. All processes with the same user_id will share the same user_cell. The management of the user cell is the application designer's responsibility.

When `AIFGLOBACQ` is called, it returns the address of the newly allocated object in the user_cell. If any value is in the user_cell prior to calling `AIFGLOBACQ` it is the user's responsibility to save the value. Likewise, if the user_cell is modified using `AIFGLOBPUT`, the application designer must save the value returned by `AIFGLOBACQ`. The value returned in the user_cell when the object was first allocated using `AIFGLOBACQ` must be placed back in the user_cell with `AIFGLOBPUT` if the user_cell has been modified between calls to `AIFGLOBACQ` and `AIFGLOBREL`.

AIFGLOBREL

Releases the object in the Architected Interface Facility: Operating System internal data area associated with the specified user ID (previously created by AIFGLOBACQ).

Syntax

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

Parameters

overall_status **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: status_type (Refer to appendix B.)

user_id **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: Operating System product.

Operation Notes

If your process does not release the shared object, the object is not deleted until the system either aborts or is shut down.

AIFGLOBACQ, AIFGLOBREL, AIFGLOBGET, and AIFGLOBPUT all access the user cell. The user cell is a 64-bit data area which is located in the AIF internal area. All processes with the same user_id will share the same user_cell. The management of the user cell is the application designer's responsibility.

When AIFGLOBACQ is called, it returns the address of the newly allocated object in the user_cell. If any value is in the user_cell prior to calling AIFGLOBACQ it is the user's responsibility to save the value. Likewise, if the user_cell is modified using AIFGLOBPUT, the application designer must save the value returned by AIFGLOBACQ. The value returned in the user_cell when the object was first allocated using AIFGLOBACQ must be placed back in the user_cell with AIFGLOBPUT if the user_cell has been modified between calls to AIFGLOBACQ and AIFGLOBREL.

AIFGLOBUNLOCK

Releases the lock on the user cell in the Architected Interface Facility: Operating System internal data area obtained by `AIFGLOBLOCK`.

Syntax

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

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

user_id **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: Operating System product.

Operation Notes

None.

AIFJSGET

Returns job and/or session information.

Syntax

```
                  REC              I32A          @64A           RECA
AIFJSGET (overall_status, itemnum_array, item_array, itemstatus_array,
              REC    I32    I32
          JSNum, JSKey, user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

JSNum **record by value (optional)**

Passes the job/session number of the job or session for which information is desired.

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

Default: 0

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

Passes the job/session key returned from a call to `AIFSYSWIDEGET`. Accessing information using this key is faster than when using the job/session number.

Default: 0

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFJSGET` accepts as an input key either of the following:

- A job/session number
- A job/session key returned from `AIFSYSWIDEGET`

Use of a job/session key provides much faster access than a job/session number. If neither parameter is specified, the default is the caller's job/session.

AIFJSPUT

Modifies job and/or session information.

Syntax

```
                  REC              I32A        @64A             RECA
AIFJSPUT (overall_status, itemnum_array, item_array, itemstatus_array,
           REC    I32     I32         I32A        @64A
          JSNum, JSKey, user_id, ver_item_nums, ver_items,
                 RECA
          ver_item_statuses);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

JSNum **record by value (optional)**

Passes the job/session number of the job or session whose information is to be modified.

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

Default: 0

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

Passes the job/session key returned from a call to `AIFSYSWIDEGET`. Modifying information using this key is faster than when using the job/session number.

Default: 0

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If *n* items are being verified, element *n*+1 must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

Operation Notes

`AIFJSPUT` accepts as an input key either of the following:

- A job/session number
- A job/session key returned from `AIFSYSWIDEGET`

Use of a job/session key provides much faster access than a job/session number. If neither parameter is specified, the default is the caller's job/session.

AIFJSGET/PUT Items

The following two tables provide summary and detailed descriptions of the item numbers associated with job/session information.

Item Summary

The following table summarizes the item numbers associated with job/session information. For more detailed information about these item numbers, refer to the table of job/session information item descriptions.

Table 3-15. Job or Session Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
1001	CA16	Job name	Y	Y			
1002	I32	Job state	N	Y			
1003	B	Duplicative?	Y	Y			
1004	B	Interactive?	Y	Y			
1005	B	Quiet mode?	Y	Y			
1006	B	`$STDLIST` state	Y	Y			
1007	I32	Input priority	Y	Y	0	15	-1005
1008	I32	Output priority	Y	Y	0	14	-1008
1009	CA16	User name	N	Y			
1010	CA16	Group name	N	Y			
1011	CA16	Account name	N	Y			
1012	I32	Input device	N	Y			
1013	JSDev_type	Output device	Y	Y			
1014	I32	Start date	Y	Y			
1015	I32	Start time	Y	Y			
1016	I32	Execution priority	Y	Y	100	250	-1009
1017	I32	JSMAIN PIN	N	Y			
1018	I32	CI PIN	N	Y			
1019	I32	CPU limit	Y	Y	-1	32767	-1010
1020	B	Spooled?	N	Y			
1021	B	Restart?	Y	Y			
1022	B	Numbered job?	N	Y			
1023	B	Programmatic session?	N	Y			
1024	I32	Max account job priority	N	Y			
1025	I32	Account security	Y	Y			
1026	I32	Group security	Y	Y			
1027	CA16	Home group name	N	Y			
1028	I32	CPU count	N	Y			
1029	I32	Directory CPU count	N	Y			
1030	I32	Account local attributes	Y	Y			

Table 3-15. Job or Session Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
1031	I32	User capabilities	Y	Y			
1032	I32	General resource capabilities	Y	Y			
1033	I32	# creations	N	Y			
1034	BA96	Allow mask	Y	Y			
1035	Longint_type	Logon timestamp	N	Y			
1036	I32	CI time out	Y	Y			
1037	JSNum_type	Job/session number	N	Y			
1038	I32	Job wait index	N	N			
1039	B	Session?	N	Y			
1040	B	Network services?	N	Y			
1043	CA16	HP DTC Portid	N	Y			
1044	REC	Job submitter job/session number	N	Y			
1045	CA16	Job submitter job/session name	N	Y			
1046	CA16	Job submitter user name	N	Y			
1047	CA16	Job submitter account name	N	Y			
1048	I32	Job submitter logical device	N	Y			
1049	I32	Job submitter session/job introduction date	N	Y			
1050	I32	Job submitter session/job introduction time	N	Y			

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with job/session information.

Table 3-16. Job or Session Information Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1001	**Job name (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the identifier given to a job or session. It must be left-justified, all capitals, and padded with blanks. All blanks represent a job or session that does not have a job name. Only the first eight characters may be changed using `AIFJSPUT`.
1002	**Job state (I32) Put: No; Verify: Yes; Release 3.0** Returns the current state of the job or session. If a job or session is in the `EXEC*` state, there is no guarantee that any of the values returned by the AIF are valid. Values and their meanings are: 1 Introduced (`INTRO`) 2 Executing (`EXEC`) 3 Terminating (`TERM`) 4 Suspended (`SUSP`) 32 Waiting (`WAIT`) 40 Error (`ERROR`) 48 Initializing (`EXEC*`) 56 Scheduled (`SCHED`)
1003	**Duplicative? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the duplicative status of the job or session. True when all input operations for a job or session are echoed to a corresponding device without intervention by the operating system software.
1004	**Interactive? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the interactive status of the job or session. True when human intervention is required for all input operations for a job or session.
1005	**Quiet mode? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the mode status of the job or session. True when the job or session is in quiet mode. While in quiet mode, the job or session will not receive `TELL` messages.
1006	**$STDLIST state (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the `$STDLIST` final disposition. True when a `SET STDLIST=DELETE` is invoked for a job (`$STDLIST` is deleted after job termination). False when `$STDLIST` is to be saved after job termination.
1007	**Input priority (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current input priority (`INPRI`) of the job. When a job's `INPRI` is higher than the system `JOBFENCE`, the system allows the job to execute. The input priority should be a value in the range 0..15. The value 15 is equivalent to using the `;HIPRI` option of the `JOB` command.

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1008	**Output priority (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current output priority (`OUTPRI`) of the job. When a job's `OUTPRI` is higher than the outfence of the output device, the spool file that is associated with the `$STDLIST` for that job is sent to the device. The output should be a value in the range 0..14.
1009	**User name (CA16) Put: No; Verify: Yes; Release 3.0** Returns the name of the user that the job or session is logged on to. It is left-justified and padded with blanks.
1010	**Group name (CA16) Put: No; Verify: Yes; Release 3.0** Returns the name of the group that the job or session is logged on to. This is left-justified and padded with blanks.
1011	**Account name (CA16) Put: No; Verify: Yes; Release 3.0** Returns the name of the account that the job or session is logged on to. This is left-justified and padded with blanks.
1012	**Input device (I32) Put: No; Verify: Yes; Release 3.0** Returns the LDEV number associated with `$STDIN` for this job or session.
1013	**Output device (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies a record that has two fields, a boolean and an integer. If the boolean is true, the integer contains a DCT index; otherwise, it contains the LDEV number of the output device for the job or session. Do not change the output device for sessions.
1014	**Start date (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the date that the job or session first logged on. If it is for a scheduled job, it is the date that the job is scheduled to start. Start date is of the following format: Bits (0:16) Start date (unused) Bits (16:7) Start date (Year after 1900) Bits (23:9) Start date (Day of Year) Do not change the start date of a job in the `SCHED` state because a change to one job may impact all jobs in the `SCHED` state.

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1015	**Start time (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the time that the job or session first logged on. If it is for a scheduled job, it is the time that the job is scheduled to start. It is of the format: Bits (0:8) Start time (Hour of Day) Bits (8:8) Start time (Minute of Hour) Bits (16:8) Start time (Second of Minute) Bits (24:8) Start time (Tenth of Second) Do not change the start time of a job in the `SCHED` state because a change to one job may impact all jobs in the `SCHED` state.
1016	**Executing priority (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies a priority that translates to the base of the queue that the job or session is logged on to. Values and their associated queues areas follows: 100 BS queue 150 CS queue 200 DS queue 250 ES queue
1017	**JSMain PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the process identification number (PIN) of the JSMain process for the job or session. A zero is returned if the job or session is in a wait state.
1018	**CI PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the process identification number (PIN) of the command interpreter for the job or session. Not valid for jobs in the `WAIT` state.
1019	**CPU limit (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the CPU time limit for the job or session. A value of -1 is equivalent to no CPU limit, the default for any job or session. Both the `HELLO` and `JOB` commands have a `TIME` parameter for changing this to a number from 1 to 32,767. This value is in seconds.
1020	**Spooled? (B) Put: No; Verify: Yes; Release 3.0** Returns or modifies the spooled state of a `$STDIN`. True when `$STDIN` for a job is a spooled device. Because there are no hot jobs on MPE/iX, this item should always remain true.
1021	**Restart? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the `RESTART` status of a job. True when the `RESTART` option was specified for a job. If the system goes down before a job with `RESTART` completes, that job is automatically rescheduled when the system comes back up.
1022	**Numbered job? (B) Put: No; Verify: Yes; Release 3.0** Returns or modifies whether the text file containing the job is numbered. True when the actual text file containing the job is numbered.

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1023	**Programmatic session? (B) Put: No; Verify: Yes; Release 3.0** Returns or modifies the programmatic session status of a session. True when a session is a programmatic session (created using the `STARTSESS` command or the `STARTSESS` intrinsic).
1024	**Maximum account job priority (I32) Put: No; Verify: Yes; Release 3.0** Returns or modifies a priority that is the maximum allowed for the account that the job or session is logged on to. The maximum priority for an account is specified by using the `MAXPRI` parameter of the `NEWACCT` and `ALTACCT` command. Not valid for jobs in the `WAIT` state. The values and their associated queues are as follows: 100 BS queue 150 CS queue 200 DS queue 250 ES queue
1025	**Account security (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the security mask for the account that the job or session is logged on to. The account security mask can also be set using the `ALTSEC` command. Not valid for jobs in the `WAIT` state. The bits of the mask have the following meanings: Bits (0:20) Unused Bit (20:1) Read any Bit (21:1) Read account user Bit (22:1) Append any Bit (23:1) Append account user Bit (24:1) Write any Bit (25:1) Write account user Bit (26:1) Lock any Bit (27:1) Lock account user Bit (28:1) Execute any Bit (29:1) Execute account user Bits (30:2) Unused

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1026	**Group security (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the security mask for the group that the job or session is logged on to. The group security mask can also be set using the `ALTSEC` command. Not valid for jobs in the `WAIT` state. The bits of the mask have the following meanings: Bits (0:2) Unused Bit (2:1) Read any Bit (3:1) Read account user Bit (4:1) Read account librarian Bit (5:1) Read group user Bit (6:1) Read group librarian Bit (7:1) Append any Bit (8:1) Append account user Bit (9:1) Append account librarian Bit (10:1) Append group user Bit (11:1) Append group librarian Bit (12:1) Write any Bit (13:1) Write account user Bit (14:1) Write account librarian Bit (15:1) Write group user Bit (16:1) Write group librarian Bit (17:1) Lock any Bit (18:1) Lock account user Bit (19:1) Lock account librarian Bit (20:1) Lock group user Bit (21:1) Lock group librarian Bit (22:1) Execute any Bit (23:1) Execute account user Bit (24:1) Execute account librarian Bit (25:1) Execute group user Bit (26:1) Execute group librarian Bit (27:1) Save any Bit (28:1) Save account user Bit (29:1) Save account librarian Bit (30:1) Save group user Bit (31:1) Save group librarian

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1027	**Home group (CA16) Put: No; Verify: Yes; Release 3.0** Returns the name of the home group for the user that the job or session is logged on to. This is left-justified and padded with blanks. Not valid for jobs in the `WAIT` state.
1028	**CPU count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of milliseconds of CPU time used by processes within the job or session that have already died. Not valid for jobs in the `WAIT` state.
1029	**Directory CPU count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of seconds of CPU time that the job or session has already been charged for as of the last `CHGROUP` command. Not valid for jobs in the `WAIT` state.
1030	**Account local attributes (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the account local attributes for the account that the job or session is currently logged on to. These attributes are an extension of MPE/iX security and are not required. Their meaning is user defined, although the first 16 bits are unused. Not valid for jobs in the `WAIT` state.
1031	**User capabilities (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the user capability mask for the job or session. Not valid for jobs in the `WAIT` state. Mask bits and their meanings are as follows: Bits (0:16) Unused Bit (16:1) System manager Bit (17:1) Account manager Bit (18:1) Account librarian Bit (19:1) Group librarian Bit (20:1) Diagnostician Bit (21:1) System supervisor Bit (22:1) Create volume sets Bit (23:1) Use private volumes Bit (24:1) Use user logging Bit (25:1) Unused Bit (26:1) Programmatic sess Bit (27:1) Network administrator Bit (28:1) Node manager Bit (29:1) Use comm subsystem Bit (30:1) Non-shareable device Bit (31:1) Save files

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1032	**General resource capabilities (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the general resources capability mask for the job or session. This mask contains the general resource capabilities for the user that the job or session is logged on to. Not valid for jobs in the `WAIT` state. Mask bits and their meanings are as follows: Bits (0:23) Unused Bit (23:1) Batch access Bit (24:1) Interactive access Bit (25:1) Privileged mode Bit (26:2) Unused Bit (28:1) Multiple RINs Bit (29:1) Unused Bit (30:1) Extra data segment Bit (31:1) Process handling
1033	**Number of creations (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of process creations performed by the job or session. This number does not include the command interpreter, and it is a count that is kept for the life of the job or session. Not valid for jobs in the `WAIT` state.

1034	**Allow mask (BA96) Put: Yes; Verify: Yes; Release 3.0**
	
	Returns or modifies the commands allowed for this session in a packed array of 96 booleans. True is returned if the command is allowed for the job or session. Not valid for jobs in the `WAIT` state. The commands and their array locations are as follows:
	
	`ABORTIO = 1` `DELETESPOOLFILE = 19` `LDISMOUNT = 37`
	`ACCEPT = 2` `DISALLOW = 20` `MRJECONTROL = 38`
	`DOWN = 3` `JOBFENCE = 21` `JOBSECURITY = 39`
	`GIVE = 4` `LIMIT = 22` `DOWNLOAD = 40`
	`HEADOFF = 5` `STOPSPOOL = 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 CONTROL = 45`
	`TAKE = 10` `RESUMESPOOL = 28` `SHOWCOM = 46`
	`UP = 11` `STREAMS = 29` `OPENQ = 47`
	`MPLINE = 12` `CONSOLE = 30` `SHUTQ = 48`
	`DSCONTROL = 13` `WARN = 31` `DISCRPS = 49`
	`ABORTJOB = 14` `WELCOME = 32` `VSRESERVESYS = 50`
	`ALLOW = 15` `MON = 33` `VSRELEASESYS = 51`
	`ALTSPOOLFILE = 16` `MOFF = 34` `VSCLOSE = 52`
	`ALTJOB = 17` `VMOUNT = 35` `VSOPEN = 53`
	`BREAKJOB = 18` `LMOUNT = 36` `SPOOLER = 54`
	`unused = 55..96`

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1035	**Logon time stamp (I64) Put: No; Verify: Yes; Release 3.0** Returns the time stamp for the time that the job or session logged on. The value is in microseconds. Not valid for jobs in the `WAIT` state.
1036	**CI time out (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of minutes that the command interpreter waits for a response before logging the session off. If this value is 0, the CI does not log off the session. This value is the same as the CI variable `HPTIMEOUT`. Not valid for jobs in the `WAIT` state.
1037	**Job/Session number (I32) Put: No; Verify: Yes; Release 3.0** Returns the job/session number for the specified job or session, in the following form: Bits (0:2) Job or Session? (1 = Session, 2 = Job) Bits (2:14) Number Bits (16:16) Extension
1038	**Job wait index (I32) Put: No; Verify: No; Release 3.0** Returns the index of the job in the wait queue, where the job with an index of 1 is the next to execute. Valid only for jobs in the `WAIT` state.
1039	**Session? (B) Put: No; Verify: Yes; Release 3.0** Returns true if a session and false if a job.
1040	**Network services? (B) Put: No; Verify: Yes; Release 3.0** Returns true when a job or session has a dsline open.
1041	**Private? (B) Put: Yes; Verify: Yes; Release 4.0** Returns or modifies the PRIVATE option for the file $STDLIST. This option will only take affect for jobs in the introduced, scheduled, and waitstate before $STDLIST is opened. This item should not be used with item 1042 since PRIVATE spoolfiles may not be SPSAVEd.
1042	**SPSAVE? (B) Put: Yes; Verify: Yes; Release 4.0** Returns or modifies the SPSAVE option for the job output $STDLIST file. When true, the spoolfile is not purged after the last copy has been printed. This option will only take affect for jobs in the introduced, scheduled, and waitstate before $STDLIST is opened. This item should not be used with item 1041.
1043	**HP DTC Portid (CA16) Put: No; Verify: Yes; Release 4.0** Returns the hpdtcportid system variable in the SHOWVAR format. The format of hpdtcportid is DTC LAN station address followed by SIC and port numbers (for example, 080009001111 0002).

Table 3-16. Job or Session Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
1044	**Job submitter job/session number (REC) Put: No; Verify: Yes; Release 4.0** Returns submitter information for both jobs and sessions. The job/session number is returned for the user who streamed the job or created the session. For system processes the job/session number returned is 0. Record type: jsnum_type (Refer to appendix B).
1045	**Job submitter job/session name (CA16) Put: No; Verify: Yes; Release 4.0** Returns the identifier name given to the submitter job or session. This is left-justified and padded with blanks. When the submitter is a system process, the name returned is blank.
1046	**Job submitter user name (CA16) Put: No; Verify: Yes; Release 4.0** Returns the name of the user that streamed the job or started the session. This is left-justified and padded with blanks. When the submitter is a system process, the user name is MANAGER from MANAGER.SYS.
1047	**Job submitter account name (CA16) Put: No; Verify: Yes; Release 4.0** Returns the name of the account the submitter was logged on to when they streamed the job or started the session. When the submitter is a system process, the account name is SYS for MANAGER.SYS.
1048	**Job submitter logical device (I32) Put: No; Verify: Yes; Release 4.0** Returns the ldev number of the submitter. When the submitter is a system process, ldev 20 is returned.
1049	**Job submitter session/job introduction date (I32) Put: No;Verify: Yes; Release 4.0** Returns the date the STREAM of STARTSESS request was issued. Bits (0:16) Submitter Date (Unused) Bits (16:7) Submitter Date (Year after 1900) Bits (23:9) Submitter Date (Day of Year)
1050	**Job submitter session/job introduction time (I32) Put: No; Verify: Yes; Release 4.0** Returns the time the submitter issued the `STREAM` or `STARTSESS` command.

AIFKSMCREATE

Returns a raw KSAM/XL file structure based on the file-specific information passed in the user buffer.

Syntax

```
   I32                                REC        A     I32
filenum := AIFKSMCREATE (overall_status, buffer, bytes,
                 I32          A          A
             user_id, group_name, acct_name,
                 CA36         B          I16
             creator,    old_date,   dev_num,
                 A            A          A
             vol_class, vol_name, vol_set_name,
                 REC
             directory, file_name);
```

Functional Return

filenum — **32-bit signed integer by reference (required)**

Returns an integer value used to identify the opened file in subsequent AIF VVVcalls.

Parameters

overall_status — **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. Refer to appendix A for meanings of status values.

buffer — **character array by reference (required)**

An array containing all of the information necessary to duplicate a KSAM/XL file. The information including the KSAM/XL control block was obtained from an `AIFKSMREAD`. It is used by `AIFKSMCREATE` to create a KSAM/XL file. The information includes the file label, file label extension, the KSAM/XL control block, and a HFS pathname for files in the hierarchical directory.

The minimum buffer size is 10,240 bytes.

bytes — **32-bit signed integer by value (required)**

An integer specifying the length of the information 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: Operating System product.

group_name

Character array by reference (optional)

Specifies an existing group where the KSAM/XL file will be created. If this parameter is not provided, the default group is where the KSAM/XL file being read resides.

Array type: `mpe_name_type` (Refer to appendix B.)

acct_name

Character array by reference (optional)

Specifies an existing account where the KSAM/XL file will be created. If this parameter is not provided, the default account is where the existing KSAM/XL file resides.

Array type: `mpe_name_type` (Refer to appendix B.)

creator

Character array by reference (optional)

Specifies the creator of the file being created. If this parameter is not provided, the creator in the file label of the existing KSAM/XL file is used.

Note that with the introduction of the hierarchical file system, the creator concept has been replaced with the concept of the file owner, which includes both the user and account name; therefore, the format of this parameter has changed to USER.ACCOUNT (padded with blanks). The name is also not upshifted.

Array type: `mpe_name_type` (Refer to appendix B.)

old_date

Boolean by value (optional)

By specifying the *old_date* to be true, the original modification and last access dates in the file label are retained. Default is false.

ldev_num

Short integer by reference (optional)

Specifies the logical device number where the KSAM/XL file will be created.

vol_class **Character array by reference (optional)**

Specifies a volume class name on which the KSAM/XL file will be created.

Array type: `t_vol_class_name` (Refer to appendix B.)

vol_name **Character array by reference (optional)**

Specifies the volume class on which the KSAM/XL file will be created.

Array type: `t_volume_name` (Refer to appendix B.)

vol_set_name **Character array by reference (optional)**

Specifies the volume set on which the KSAM/XL file will be created.

Array type: `t_vol_set_name` (Refer to appendix B.)

directory **record by reference (optional)**

Passes the absolute pathname of the directory where the KSAM/XL file will be created. If this parameter is specified, then the *group_name* and *acct_name* parameters will be ignored. An example of a valid pathname for this parameter is `/SYSUTIL/MPEXL/tools_directory/`.

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

file_name **record by reference (optional)**

Passes the new name of the KSAM/XL file to be created. If this name is specified along with *group_name* and *acct_name*, then the file is created in the MPE domain. If *directory* is specified then a POSIX file is created. A *directory* option overrides the *group_name* and *acct_name*.

If *file_name* is specified and neither *directory*, nor *group_name* and *acct_name* are specified, then the file is created in the same domain as the original file with a new name.

Record type: `pathname_type` (Refer to appendix B)

Operation Notes

The `AIFKSMCREATE` call creates a raw KSAM/XL file using the information contained in the buffer. The first `AIFKSMREAD` of a KSAM/XL file must be called before `AIFKSMCREATE` to obtain the necessary information in ***buffer*** for creating the file. At the end of `AIFKSMCREATE`, the data pointer is positioned to the next physical byte to be written, or at the end of the file if the entire file has been completely written.

AIFKSMREAD

Sequentially reads a physical block of user-specified size from a KSAM/XL file.

Syntax

```
 I32                          REC          I32       A
lgth := AIFKSMREAD (overall_status, filenum, buffer,
              I32      I32
            bytes, user_id);
```

Functional Return

lgth — **32-bit signed integer by reference (required)**

Returns a positive integer value indicating the length of the information transferred.

Parameters

overall_status — **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. Refer to appendix A for meanings of status values.

filenum — **32-bit signed integer by value (required)**

An identifier supplying the file number of the file to be read.

buffer — **character array by reference (required)**

An array to hold a physical block of the KSAM/XL file. The content in the buffer returned from the first `AIFKSMREAD` after the KSAM/XL file is opened is used by `AIFKSMCREATE` to duplicate a KSAM/XL file. The contents returned from subsequent `AIFKSMREAD`s are used by `AIFKSMWRITE`s to copy to the new KSAM/XL file.

The minimum buffer size is 10,240 bytes.

bytes — **32-bit signed integer by value (required)**

A positive integer specifying the number of bytes to be transferred. If this value is zero, no transfer occurs. If ***bytes*** is larger than the remaining physical block size, transfer is limited to the length up to EOF.

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: Operating System product.

Operation Notes

The AIFKSMREAD call reads a block from a KSAM/XL file in its physical sequence. The KSAM/XL file must have been opened in copy mode with MR and NOBUF options and read-only access in order to call this procedure. The first AIFKSMREAD after the file is opened transfers all of the necessary information required by AIFKSMCREATE. The buffer size and the bytes count must be large enough to hold the information to be passed to AIFKSMCREATE. The minimum size required may vary depending on the size of user labels. In the case where no user label exists in the file, the minimum size required is 10240 bytes. It is recommended that the buffer size and the bytes count be multiples of 4096 and 65536 bytes (16 pages). At the end of AIFKSMREAD procedure, the data pointer is positioned to the byte following the last byte being read. AIFKSMREAD transfers only the in-use areas and ignores the unused area in the KSAM/XL file. An end-of-file status is returned if AIFKSMREAD is called after the last byte of the file is transferred.

The AIFKSMREAD procedure returns a positive integer value to ***lgth*** showing the length of the information transferred. Both ***lgth*** and ***bytes*** in the AIFKSMREAD call must be positive numbers representing bytes counts.

AIFKSMWRITE

Sequentially writes a block of data to a KSAM/XL file in the physical order.

Syntax

```
                    REC           I32A       A
AIFKSMWRITE (overall_status, filenum, buffer,
                I32      I32
               bytes, user_id);
```

Parameters

overall_status **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. Refer to appendix A for meanings of status values.

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

An identifier supplying the file number of the file to be written.

buffer **character array by reference (required)**

Contains the block of data to be written. The content of the block was obtained from an AIFKSMREAD.

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

An integer specifying the number of bytes to be transferred. If the value is zero, no transfer occurs. If ***bytes*** is larger than the remaining bytes of the file, only the number of bytes up to the EOF are written to the file.

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

The AIFKSMWRITE call writes a block of data to the KSAM/XL file. The contents are contained in the array buffer, which was obtained from an AIFKSMREAD call. AIFKSMWRITE writes the file in its physical sequence. It writes the indexes of the file, skips the unused index area, then writes the data records. Following the execution of AIFKSMWRITE, the data pointer is positioned to the next physical byte to be written, or at the end-of-file if the last byte of the file has already been written.

AIFMOALLOCATE

Allocates a magneto-optical media drive.

Syntax

```
                    REC        I32
AIFMOALLOCATE(overall_status, ldev,
                       I32A         @64A
                  itemnum_array, item_array,
                        RECA        I32
                  itemstatus_array, user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

This parameter returns the logical device number (ldev) of the optical drive allocated.

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the *item_array* parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the *itemnum_array* parameter, each of which is a globalanyptr that points to the appropriate type needed by each particular item number. The value used for each option is taken from, or returned to, the location pointed to by the globalanyptr in this array. When this parameter

is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr` (Refer to Appendix B.)

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with the specific items, an error status is placed in the corresponding element of this array for each item with an error. The *overall_status* parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and the *item_array* parameters.

A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFMOALLOCATE` provides a way to allocate a magneto-optical media drive for dedicated use. Attempting to mount media (using `AIFMOMOUNT`) will fail with an error if the specified drive is not currently allocated.

When this AIF is used, the optical drive is allocated using the pin of the calling process. A drive that is allocated must be deallocated using the same pin. If you are the process that performed the allocate, you can call `AIFMODEALLOCATE` without explicitly specifying the pin. In addition, if you are the process that performed the allocate, you can call `AIFMOMOUNT/DISMOUNT` and `AIFMOGET/PUT` without explicitly specifying the pin. If you are not the process that performed the allocate, attempting to deallocate, mount, dismount, etc., without passing the pin for the process that performed the allocate will fail with an error. The pin can be returned to the user through the pin item number in the call to `AIFMOALLOCATE`.

Allocating an optical drive does not prevent other processes from accessing media mounted on the allocated drive, but it does prevent other processes from dismounting the current media and mounting another piece of media, unless they know the pin used to allocate the drive. If the process performing the allocate terminates before the deallocate of the drive is performed the deallocate will occur during normal process termination clean up. That is, process termination will handle cleaning up after any outstanding 'allocates' performed through the AIFs for the terminating process.

If the input ldev item or media label item is not specified, the first unallocated optical drive will be allocated. Note, for this case, if all drives are currently allocated by other processes, an error will be returned. Also, note, if all the magneto-optical drives on the system are allocated by the calling process, and `AIFMOALLOCATE` is called without specifying an ldev item or media label item, an error will not be returned but the ldev parameter will not return any particular ldev value.

If a particular item is specified more than once in a call to this AIF, the first occurrence of it will be used. For example, if item 17101 (Pin) is passed in twice, the value for the pin of the calling process will be returned in the corresponding item array for the first index for which this item was passed and a warning will be returned in the item status array for the second index for which this item was passed (a value will not be returned in the item array for the second index). Likewise, if item 17102 (ldev) is passed in twice, the first ldev value that is passed is used to perform the allocate, and the 2nd ldev value is ignored (a warning is returned in the item status array). Note, passing in the ldev (or media label) item in more than once does NOT attempt to allocate more than drive.

When this AIF is used, no actual Autochanger or I/O operations are performed.

Refer to the Programming Example in Appendix C, or the `AIFMOMOUNT` Operation Notes for more information.

AIFMOALLOCATE Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with `AIFMOALLOCATE`.

Table 3-17. AIFMOALLOCATE Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
17101	**Pin (I32); Release 5.0** Returns the pin of the calling process. The pin can be used with other magneto-optical AIF's for verification of ownership for an optical media drive. Default: Pin of the calling process
17102	**Input ldev (I32); Release 5.0** Passes the logical device number (ldev) of the optical drive to allocate. This item is not valid if the media label item is specified. Default: nil
17103	**Media label (REC) Release 5.0** Passes a media label record. This record consists of a media name, subname1, and subname2. The media name consists of an array of 1 to 32 characters and identifies the first part of the media label. Subname1 and subname2 consist of arrays of 1 to 16 characters and identify the second and third parts of the media label. Each of the fields of this record must be left justified. When this item is passed, the first available drive which can access the specified media is allocated. Note, the specified media is not allocated, only a drive which can access the requested media is allocated. By specifying a media label, you are not required to know the optical drive's ldev numbers. The media name, subname1, and subname2 must contain either a name or the character "@". "@" indicates that this field should be ignored. For example, if the following was passed: media name = MYMEDIA subname1 = @ subname2 = @ this would lock a drive which can access any piece of media whose media name was "MYMEDIA". The following media labels would be considered matching the above passed media label: media name = MYMEDIA subname1 = SUB1 subname2 = SUB2 media name = MYMEDIA subname1 = subname2 = SUB2 This item is not valid if the input ldev item is specified. Also, it is invalid to specify a media name, subname1, or subname2 that is longer than 1 character and whose first character is the character "@". Record type: media_label_type (Refer to Appendix A) Default: nil

AIFMODEALLOCATE

Deallocates a previously allocated magneto-optical media drive.

Syntax

```
                          REC       I32        I32A
AIFMODEALLOCATE(overall_status, ldev, itemnum_array,
                          @64A         RECA          I32
                    item_array, itemstatus_array, user_id)
```

Parameters

overall_status — **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

ldev — **32-bit signed integer by value (required)**

The logical device number (ldev) of the optical drive to deallocate.

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the *item_array* parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the *itemnum_array* parameter, each of which is a globalanyptr that points to the appropriate type needed by each particular item number. The value used for each option is taken from the location pointed to by the globalanyptr in this array. When this parameter is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr` (Refer to Appendix B.)

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with the specific items, an error status is placed in the corresponding element of this array for each item with an error. The *overall_status* parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and the *item_array* parameters.

A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

AIFMODEALLOCATE provides a way to deallocate a magneto-optical drive that has been previously allocated. Once the drive is deallocated, it becomes available for other users. If the pin item is not specified, this AIF will deallocate the drive on behalf of the calling process. If the process that performed the allocate terminates before the deallocate of the drive is performed the deallocate will occur during normal process termination clean up. That is, process termination will handle cleaning up after any outstanding 'allocates' performed through the AIFs for the terminating process. Note, AIFMODEALLOCATE will succeed even though media is mounted on the allocated drive. For example, the following sequence of calls will result in a successful deallocation of a drive:

```
AIFMOALLOCATE
AIFMOMOUNT
AIFMODEALLOCATE
```

If an item is specified more than once in a call to this AIF, the first occurrence of it will be used. For example, if item 17201 (Pin) is passed in twice, the first pin value that is passed is used to perform the deallocate, and the 2nd pin value is ignored (a warning will be returned in the item status array).

Also, it should be noted that a warning or error will not be returned if a valid optical disk drive is specified and the drive is already deallocated (free).

Refer to the Programming Example in Appendix C, or the AIFMOMOUNT Operation Notes for more information.

AIFMODEALLOCATE Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with `AIFMODEALLOCATE`.

Table 3-18. AIFMODEALLOCATE Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
17201	**Pin(I32); Release 5.0** Passes the pin for the process that allocated the specified optical drive. This is used for verification of ownership for an optical media drive. If a 0 is passed, the pin of the calling process is used. Default: Pin of the calling process

AIFMODISMOUNT

Logically and physically dismounts previously mounted magneto-optical media from a magneto-optical drive.

Syntax

```
                    REC       I32        I32A
AIFMODISMOUNT(overall_status, ldev, itemnum_array,
                    @64A         RECA          I32
              item_array, itemstatus_array, user_id)
```

Parameters

overall_status — **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

ldev — **32-bit signed integer by value (required)**

The logical device number (ldev) of the optical drive to dismount.

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the item_array parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the itemnum_array parameter, each of which is a globalanyptr that points to the appropriate type needed by each particular item number. The value used for each option is taken from, or returned to, the location pointed to by the globalanyptr in this array. When this parameter is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr` (Refer to Appendix B.)

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with the specific items, an error status is placed in the corresponding element of this array for each item with an error. The *overall_status* parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and the *item_array* parameters.

A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

AIFMODISMOUNT provides a way to dismount magneto-optical media that has been previously mounted. Volume management is called to close the volume set. Closing the volume set, dismounts it from the active system volume sets. All files on the volume set must be closed in order for the dismount to succeed. When a dismount occurs, the media is removed to the original media storage slot; an unoccupied storage slot if the original storage slot is occupied; or the original mail slot if no storage slot is currently unoccupied. If the media originally came from a mail slot when it was mounted, it will be returned to that mail slot when dismounted. If the pin item is not specified an attempt is made to dismount the media on behalf of the calling process.

It is recommended that all files be closed on the volume set, prior to calling AIFMODISMOUNT. If files are still open on the volume set when AIFMODISMOUNT is called, an error is returned (-17016) and the volume set is left in a 'close pending' state. When the last file on the volume set is closed, Volume Management closes the volume set and it is left in a 'LONER' state. Note, it is NOT physically removed from the drive. In order to get it physically removed, you must allocate the drive (if you do not currently have it allocated), and perform the dismount again. Mounting another piece of media will also physically remove the media from the drive (mounting causes an implicit dismount).

If an item is specified more than once in a call to this AIF, the first occurrence of it will be used. For example, if item 17401 (Pin) is passed in twice, the first pin value that is passed is used to perform the dismount, and the 2nd pin value is ignored (a warning will be returned in the item status array).

Refer to the Programming Example in Appendix C, or the AIFMOMOUNT Operation Notes for more information.

AIFMODISMOUNT Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with AIFMODISMOUNT.

Table 3-19. AIFMODISMOUNT Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
17401	**Pin (I32) Release 5.0** Passes the pin for the process that allocated the specified optical drive. This is used for verification of ownership for an optical media drive. If 0 is passed, the pin of the calling process is used. Default: Pin of the calling process
17402	**Nowait identifier (I32) Release 5.0** When a 0 is passed specifying 'initiation', the call to this AIF will initiate the dismount but will return before it completes. A unique non-zero identifier is returned and must be used in a second call to AIFMODISMOUNT to complete the dismount request. Note, the process that 'initiates' the dismount must be the process that 'completes' the dismount. For example, the following will fail with an error: Process 1 calls AIFMODISMOUNT with the nowait item set to the value 0. Process 2 passes the pin item with the value 1 (for Process 1) and calls AIFMODISMOUNT with the nowait item set to the identifier returned from the previous AIFMODISMOUNT. The second call to AIFMODISMOUNT will fail with an error, since Process 2 did NOT 'initiate' the mount. Also, any errors that occur during the actual dismount of the media are not returned until you 'complete' the dismount. This item allows the user to initiate a dismount request and to have control returned before completion of the dismount. When the second call to AIFMODISMOUNT is made to complete the mount, that is a non-zero value for the nowait identifier is passed, the ldev parameter is ignored. In addition, any items that are passed are ignored. The maximum number of nowait requests per process is 32. That is, only 32 nowait requests can be outstanding from both AIFMOMOUNT and AIFMODISMOUNT at any one time. Default: Dismount is performed before returning to user

AIFMOGET

Returns magneto-optical disk library system information.

Syntax

```
                REC       I32       I32A
AIFMOGET(overall_status, ldev, itemnum_array,
                @64A              RECA
            item_array, itemstatus_array,
            I32     I32
            pin, user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

Passes one of the following:

- The logical device number (ldev) of the optical drive which contains mounted media for which information is to be retrieved.
- The logical device number of the autochanger for the magneto-optical device for which information is to be retrieved.

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

An array of integers where each element is an item number indicating the magneto-optical device system information to be returned to a data structure pointed to in the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is to be returned. Information and its required data type are defined by the item number passed in the corresponding element in *itemnum_array*.

Array type: `globalanyptr` (Refer to Appendix B.)

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

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

Passes the pin of the process that was used to allocate the optical media that contains the mounted media for which information is to be retrieved. This is used for verification of ownership for an optical drive. If 0 is passed, the pin of the calling process is used.

Default: 0

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

If the ldev parameter passed specifies an optical drive which contains mounted media for which information is to be retrieved, and the pin parameter is not specified, an attempt is made to retrieve information on behalf of the pin of the calling process. If the ldev parameter passed specifies an autochanger and the pin parameter is passed, the pin is ignored.

AIFMOPUT

Modifies magneto-optical disk library system information.

Syntax

```
                REC       I32          I32A
AIFMOPUT(overall_status, ldev, itemnum_array,
              @64A           RECA       I32
          item_array, itemstatus_array, pin,
              I32A           @64A
          ver_item_nums, ver_items,
                RECA          I32
          ver_item_statuses, user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

Passes one of the following:

- The logical device number (ldev) of the optical drive which contains mounted media for which information is to be modified.
- The logical device number of the autochanger for the magneto-optical device for which information is to be modified.

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

An array of integers where each element is an item number indicating the magneto-optical disk library system information to be modified. New information must be located in a data structure pointed to by the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in *itemnum_array*.

Array type: `globalanyptr` (Refer to Appendix B.)

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in *item_array*. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

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

Passes the pin of the process that was used to allocate the optical media that contains the mounted media for which information is to be modified. This is used for verification of ownership for an optical drive. If 0 is passed, the pin of the calling process is used.

Default: 0

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

An array of integers where each element is an item number indicating magneto-optical disk library system information to be verified before proceeding with the modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element n+1 must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current magneto-optical disk library system information. Information and its required data type are defined by the item number passed in the corresponding element in ver_item_nums.

Array type: `globalanyptr` (Refer to Appendix B.)

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

If the ldev parameter passed specifies an optical drive which contains mounted media for which information is to be retrieved, and the pin parameter is not specified, an attempt is made to modify information on behalf of the pin of the calling process. If the ldev parameter passed specifies an autochanger and the pin parameter is passed, the pin is ignored.

AIFMOGET/PUT Item Descriptions

The following tables provide a summary and detailed descriptions of the items associated with optical drives.

Table 3-20. AIFMOGET/PUT Item Descriptions when Idev is an Optical Drive

Item Number	Item Name (Data Type) Put; Verify; Release Description
17001	**Media Label (REC) Put: YES; Verify: YES; Release 5.0** Returns or modifies the media label. The media label is a record consisting of a media name, subname1, and subname2. The media name consists of an array of 1 to 32 characters and identifies the first part of the media label. Subname1 and subname2 consist of arrays of 1 to 16 characters and identify the second and third parts of the media label. Each of the fields of this record must be left justified. The media name, subname1, and subname2 must contain either a name or the character "@". "@" indicates that this field should be ignored. (Refer to the item description of media label in the "AIFMOALLOCATE Item Descriptions") When this item is specified, the media whose media label is to be retrieved or modified, must be mounted. Since the media name, "$SCRATCH", is a system defined name used by TurboSTORE/iX, if it is passed to `AIFMOPUT` to modify a media label, unexpected results may occur if TurboSTORE/iX attempts to use this media. Record_type: media_label_type (Refer to Appendix A)
17002	**Volume Set Name (CA8) Put: NO; Verify: YES; Release 5.0** Returns a character array containing the volume set name for the mounted optical media. When this item is specified, the media whose volume set name is to be retrieved, must be mounted.

Table 3-21. AIFMOGET/PUT Item Descriptions when ldev specifies an Autochanger

Item Number	Item Name (Data Type) Put; Verify; Release Description
17003	**Number of storage slots (I32) Put: NO; Verify: NO; Release 5.0** Returns the number of storage slots.
17004	**Number of drives (I32) Put: NO; Verify NO; Release 5.0** Returns the number of drives. The first word of the buffer will be expected to hold the size, in words, of the rest of the buffer area. The first word upon return specifies the number
17005	**List of drive ldevs (REC) Put: NO; Verify: NO; Release 5.0** Returns a list of drive ldevs. The first word of the buffer will be expected to hold the size, in words, of the rest of the buffer area. The first word upon return specifies the number of ldevs returned. Record type: drives_type (Refer to Appendix A)
17006	**Number of mail slots (I32) Put: NO; Verify: NO; Release 5.0** Returns the number of mail slots.
17007	**List of storage slot information (REC) Put: NO; Verify: NO; Release 5.0** Returns an array of information for the storage slots. The information returned includes: ■ slot number ■ media labels for the media associated with a slot ■ volume set names for the media associated with a slot ■ whether the slot contains media or not On input, the first two fields in the buffer will represent the range of storage slots to retrieve. The first field (first 4 bytes) represent the lower limit and the second field (next 4 bytes) will represent the upper limit. On output, the first field (first 4 bytes) will contain the number of storage slots for which information was actually returned. Record type: storage_slot_type (Refer to Appendix A)

AIFMOMOUNT

Physically and logically mounts magneto-optical media by loading it into a magneto-optical drive and mounting it into the file system.

Syntax

```
                    REC       I32      REC
AIFMOMOUNT(overall_status, ldev, media_label,
                     I32A          @64A          RECA
                itemnum_array, item_array, itemstatus_array,
                  I32
                user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

The logical device number (ldev) of the optical drive where the specified media should be mounted.

media_label **record by reference (required)**

Passes an optical media label for the optical media to mount. This record consists of a media name, subname1, and subname2. The media name consists of an array of 1 to 32 characters and identifies the first part of the media label. Subname1 and subname2 consist of arrays of 1 to 16 characters and identify the second and third parts of the media label. Each of the fields of this record must be left justified.

The media name, subname1, and subname2 must contain either a name or the character "@". "@" indicates that this field should be ignored. (Refer to the item description of media label in the "AIFMOALLOCATE Item Descriptions")

Record type: `media_label_type` (Refer to Appendix B.)

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the *item_array* parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the itemnum_array parameter, each of which is a globalanyptr that points to the appropriate type needed by each particular item number. The value used for each option is taken from, or returned to, the location pointed to by the globalanyptr in this array. When this parameter is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr` (Refer to Appendix B.)

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with the specific items, an error status is placed in the corresponding element of this array for each item with an error. The *overall_status* parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and the *item_array* parameters.

A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: `status_type` (Refer to Appendix B.)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFMOMOUNT` provides a way to physically mount a specific piece of magneto-optical media in a magneto-optical drive and then logically mount it in the file system. The logical mount of the magneto-optical media is implicit and it is performed automatically through this AIF.

The following sections provide a brief overview of some of the concepts related to mounting media in a magneto-optical disk library system: MOUTIL, Volume Set, and Media Label.

MOUTIL

To initially load an optical disk library system with optical media, the user must run the MOUTIL utility. Through MOUTIL, the user can load/unload media, initialize and scratch media, synchronize internal tables, and perform other housekeeping commands. Once media has been loaded, initialized, or synchronized within an optical disk library system, applications can make use of `AIFMOMOUNT` to mount media into the MPE/iX File System. Note, when initializing optical media that will be accessed through the magneto-optical AIFs, the INITMO command in MOUTIL should be used with the STORE=NO option.

Volume Set

The basic entity used by the Media Manager is a user volume. When optical media is mounted (e.g. through `AIFMOMOUNT`), each surface is treated as a master user volume with no member volumes, that is, a single user volume set. 'Mounted' refers to the Media Manager physically mounting the media by moving the media from a storage slot to a drive and then logically mounting it by causing it to go 'online', thus causing an implied VSOPEN to occur.

The volume set described here is the same as a volume set managed through Volume Management. The only differences include the following. The optical media volume set is formatted through the MOUTIL utility as opposed to the VOLUTIL utility, and it is indirectly mounted through the Media Manager. Another difference is that an optical media volume set can only consist of a single volume, the master volume. Also, the volume set name for optical media is limited to 8 alphanumeric characters ("_" and "." are not allowed) as opposed to 32 alphanumeric characters for a non-optical media volume set. This limitation exists because the Media Manager uses the volume set name to create a group, in the account HPOPTMGT, with the same name as the volume set name of the optical media. This group is created in order for the Media Manager to access media information on an optical media volume set. The file MEDINFO.(volume set name).HPOPTMGT is built on the media when it is initialized through MOUTIL and it contains optical media information used by the Media Manager. In addition, the Media Manager uses this group to bind the volume set directory to the system directory.

Once an optical media volume set is mounted, it can be treated as any user volume set on the system.

Media Label

Since optical media volume sets can only consist of a single volume, the master volume, the Media Label can be used to logically relate optical media volume sets (optical media surfaces) when multiple surfaces are required. The media label consists of three parts: a media name and two subnames. The media label is user definable and can be updated using `AIFMOPUT`. Once the media label has been initialized through MOUTIL, it can be modified in order to manage and logically relate optical media. Media labels do not have to be unique within a particular library system or across library systems. One example of logically relating optical media is as follows. Consider the case where you would want to relate 3 optical media volume sets. You could name them in the following way:

media name = MYMEDIA subname1 = SET1 subname2 = DEC92

media name = MYMEDIA subname1 = SET2 subname2 = JAN93

media name = MYMEDIA subname1 = SET3 subname2 = FEB93

Though the media label allows you to logically relate optical media, it does not provide you with the same capability as a multiple volume, volume set (that is, files will not span optical media).

AIFMOMOUNT Notes

Attempting to mount media will fail with an error if the specified drive is not currently allocated (using `AIFMOALLOCATE`). If media is mounted in a drive and no one is accessing the media (that is, no files are open), the media can be dismounted and another media mounted. If the pin item is not specified an attempt is made to mount the media on behalf of the calling process. If the process who allocated the drive where the media was mounted terminates before the dismount is performed, before the deallocate is performed, the dismount and deallocation will occur during normal process termination clean up.

If a particular item is specified more than once in a call to this AIF, the first occurrence of it will be used. For example, if item 17303 (volume set) is passed in twice. The volume set name will be returned in the corresponding item array for the first index for which this item was passed. A warning will be returned in the item status array for the second index for which this item was passed. A value will not be returned in the item array for the second index. Likewise, if item 17302 (prompt for media) is passed in twice. The first value that is passed is used to perform the mount. The second prompt for media value is ignored. A warning is returned in the item status array.

AIFMOMOUNT Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with `AIFMOMOUNT`.

Table 3-22. AIFMOMOUNT Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
17301	**Pin (I32) Release 5.0** Passes the pin for the process that allocated the specified optical drive. This is used for verification of ownership for an optical media drive. If 0 is passed, the pin of the calling process is used. Default: Pin of the calling process
17302	**Prompt for Media (I32) Release 5.0** Passes an indicator specifying the media prompting when media is in use or could not be found. The valid inputs are as follows: 1 - no prompting 2 - prompt media not found 3 - prompt media in use If "no prompting" is specified then no system console message is issued to mount the media which could not be found and an error will be returned. If "prompt media not found" is specified then the user will be prompted with a system console message to mount the requested media if the media does not currently exist within the library the specified drive has access to. If "prompt media in use" is specified then the user will be prompted with a system console message to mount media with the same name if the specified media is currently being used. This item prints out a system console message and waits for an operator reply. If the reply is outstanding and is not processed immediately, any subsequent magneto-optical AIF calls may appear to be hung waiting for the first reply to be processed and the mount to complete. For similar reasons, caution should be used when using this item with the nowait item. Default: 1
17303	**Volume Set Name (CA8) Release 5.0** Returns a character array that identifies the volume set name of the media mounted. This item is not valid if the nowait identifier item specifies an 'initiation' request.

Table 3-22. AIFMOMOUNT Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
17304	**Nowait identifier (I32) Release 5.0** This item allows the user to initiate a mount request and to have control returned before completion of the mount. When a 0 is passed specifying 'initiation', the call to this AIF will initiate the mount but will return before it completes. A unique non-zero identifier is returned and must be used in a second call to `AIFMOMOUNT` to complete the mount request. Note, the process that 'initiates' the mount must be the process that 'completes' the mount. For example, the following will fail with an error: Process 1 calls `AIFMOMOUNT` with the nowait item set to the value 0. Process 2 passes the pin item with the value 1 (for Process 1) and calls `AIFMOMOUNT` with the nowait item set to the identifier returned from the previous `AIFMOMOUNT`. The second call to `AIFMOMOUNT` will fail with an error, since Process 2 did NOT 'initiate' the mount. Also, any errors that occur during the actual mounting of the media are not returned until you 'complete' the mount. When the second call to `AIFMOMOUNT` is made to complete the mount, using a non-zero value for the nowait identifier, the ldev and media label parameters are ignored. In addition, any items that are passed are ignored. The maximum number of nowait requests per process is 32. That is, only 32 nowait requests can be outstanding from both `AIFMOMOUNT` and `AIFMODISMOUNT` at any one time. Default: Mount is performed before returning to user

AIFPORTCLOSE

Removes a connection to a port opened by a call to `AIFPORTOPEN`.

Syntax

```
                    REC         I32         I32
AIFPORTCLOSE (overall_status, port_id, access_mode);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

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

The port ID of the port to close (the identifier returned from a successful call to `AIFPORTOPEN`).

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

Individual access modes may be closed separately. This parameter specifies the mode to close with this call to `AIFPORTCLOSE`. The access need not be the same as used in the `AIFPORTOPEN`. If the calling process does not have the port open for the specified access, the close is ignored for that access. If not passed, the port is closed for all access modes. Values and their meanings are as follows:

1	Receive access
2	Send access
3	Both receive and send access

Default: all access modes

Operation Notes

For every AIFPORTOPEN performed during the life of a process, a corresponding AIFPORTCLOSE should be performed. In the event of a process abort, or if the process neglects to call AIFPORTCLOSE for any or all of the ports it has open, the ports are closed automatically during the process termination sequence. If the port is not specified to be a permanent port by the last process to open the port, it is destroyed when the last opener closes the port. If the port is a permanent port, it remains after the last process closes it.

Asynchronous ports are always temporary and have only a single receiver (the creator); therefore, when the creating process terminates or calls AIFPORTCLOSE with receive access, subsequent sends to the port return an error since the receiver no longer exists. See "Operation Notes" on AIFPORTOPEN for more details on asynchronous ports.

AIFPORTINT

Allows the user to change the interrupt handler state of one or more asynchronous ports. The caller of this routine must be the receiver of the port.

Syntax

```
                  REC          I32A       BA         BA
AIFPORTINT (overall_status, port_list, newstates, oldstates);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

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

Passes an array of integers that contain a list of port IDs whose handlers will be enabled or disabled. This array MUST be terminated with a zero. Each port ID must be an asynchronous port that has been created by the caller. If any port ID is invalid, then NONE of the ports will have their interrupt state changed.

newstates **Boolean array by reference (required)**

An array of Booleans that specify the new port interrupt handling states that are the result of this call. A value of TRUE enables interrupt handling for the corresponding port ID in the ***port_list***, while FALSE disables interrupt handling on a port. This array must contain as many elements as are contained in ***port_list***.

oldstates **Boolean array by reference (optional)**

An array of booleans, that upon return from a successful call contains a value of TRUE for each port in ***port_list*** that had interrupts enabled prior to this call, and FALSE for each port that had interrupts disabled. This array must contain as many elements as are contained in ***port_list***. If the call fails because of an invalid port ID and this array was passed, a value of FALSE is returned for each port that was not previously created by the caller.

Default: nil

Operation Notes

The `AIFPORTINT` routine has been provided to allow a user to enable or disable interrupt handling when a message arrives on an AIF port. When interrupt handling is disabled on a port, calls to the interrupt handling routine are delayed until interrupt handling is reenabled with the `AIFPORTINT` routine.

AIFPORTOPEN

Creates and/or opens a port. The port can be opened to allow for the asynchronous receipt of incoming messages by enabling a user specified handler.

Syntax

```
 I32                          REC          CA16         CA16
port_id := AIFPORTOPEN (overall_status, port_name, port_password,
                             I32      I32       I32A          @64A
                        access_mode, user_id, itemnum_array, item_array,
                                RECA
                        itemstatus_array);
```

Functional Return

port_id — **32-bit signed integer by reference (required)**

Returns a unique identifier, a port ID, to be used with other port AIFs to manage the opened port. The maximum number of open AIF ports is 2048.

If the call to `AIFPORTOPEN` is unsuccessful, ***port_id*** is undefined. Check ***overall_status*** to determine which error occurred.

Parameters

overall_status — **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

port_name **character array by reference (required)**

Passes name used to identify this particular port. This name must be unique across the entire system. It should be padded on the right with blanks if it is fewer than 16 characters. The name will be upshifted, so it is not case sensitive.

If a totally blank port name is specified, a unique name is established, a port with that name is created, and the name is returned in the ***port_name*** parameter.

Array type: `pac16` (Refer to appendix B.)

port_password **character array by reference (required)**

Passes a password to associate with the port being opened. This password can be up to 16 characters in length. It should be padded on the right with blanks if it is fewer than 16 characters. The password will be upshifted, so it is not case sensitive. The first open of a port establishes the password, which must be matched by all subsequent opens.

Array type: `pac16` (Refer to appendix B.)

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

Passes a value determining the access mode for the port being opened.

Values and their meanings are as follows:

1	Receive access
2	Send access
3	Both receive and send access

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the *item_array* parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the *itemnum_array* parameter, each of which is a `globalanyptr` that points to the appropriate type needed by each particular item number. The value used for each option is taken from, or returned to, the location pointed to by the `globalanyptr` in this array. When this parameter is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr`

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with specific items, an error status is placed in the corresponding element of this array for each item with an error. The overall status parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and *item_array* parameters.

A non-zero value indicates an error, but a valid option does not set the value to zero, so this array should be initialized to all zeros before making the call.

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

Default: nil

Operation Notes

The AIF Port Facility is an application interface that provides a fast means of interprocess communication by sending messages from one process to another. Messages can be received in a synchronous or asynchronous fashion. The ability to receive messages asynchronously is determined when the port is created.

The remaining notes will reference details from the AIFPORTOPEN item descriptions (Table 3-16). Please review the item descriptions before reading further.

Opening An AIF Port

The first time that AIFPORTOPEN is called for a named port that does not exist, it is created by default. If the named port already exists, it is opened. AIFPORTOPEN returns an integer value that must be supplied to all other port AIFs to identify the port being referenced. The default AIFPORTOPEN creates the port as temporary and does not allow for the asynchronous receipt of messages.

An *asynchronous port* is a port that provides the capability of interrupting the creator upon receipt of a message and transfers control to a user specified handler. To create an asynchronous port specific items must be passed to the AIFPORTOPEN routine. The following example illustrates the creation of an asynchronous port.

```
readln (user_id);
portname          := 'aifport1        ';
portpass          := 'aifpass1        ';
accessmode        := 1;          { receive access }
itemnum_ports     := Init_Itemnum_Array; { zero array }
item_ports        := Init_Item_Array;
item_status_ports := Init_Item_Status_Array;
createoptions     := 2;          { create new     }
itemnum_ports [1] := 11201;
item_ports [1]    := ADDR(createoptions);
maxmsgsize        := 80;         { message size   }
itemnum_ports [2] := 11202;
item_ports [2]    := ADDR(maxmsgsize);
proc_name         := '#PORTHANDLER#';
proc_file         := '#ASYNC1#';

HPGETPROCPLABEL (proc_name, createhandler,
          overall_status, proc_file, False);
if overall_status.all <> 0 then
  ERROR_IN_CALL('HPGETPROCPLABEL',overall_status);

itemnum_ports [3] := 11206;
item_ports [3]    := addr(createhandler); { handler address }
createstate       := True;
itemnum_ports [4] := 11207;  { next element initialized to 0 }
item_ports [4]    := addr(createstate); { enabled }

portid1 := AIFPORTOPEN(overall_status, portname, portpass,
         accessmode,,itemnum_ports,
         item_ports, item_status_ports);
if overall_status.all <> 0 then
  ERROR_IN_CALL('AIFPORTOPEN',overall_status,
   item_status_ports);
```

The creator of an asynchronous port is the only process that may receive messages from this port, and must provide the handler address when opening the port. If the creating process abnormally terminates, subsequent sends to the port will return an error.

AIF ports that do NOT specify a handler at creation time, receive messages synchronously and allow multiple receivers. In addition, synchronous ports can be permanent, however, asynchronous ports are always temporary.

Handlers

Handlers for asynchronous ports must be coded to certain conventions in order to function properly. The address of the handler can be acquired by calling the intrinsic `HPGETPROCLABEL` as shown in the previous example.

When defining the handler routine, the calling sequence must have one parameter. This parameter will contain the portid of the AIF port which received the asynchronous message.

In Pascal/iX, a handler is declared as follows:

procedure INT_HANDLER (port_id : integer);

In C/iX, it would be:

void INT_HANDLER (int portid)

Handlers should do only what is absolutely necessary. It is NOT a good idea to do the `AIFPORTRECEIVE` using the "all ports" option, as this can result in unnecessary delays. When using item 11003 of `AIFPORTRECEIVE` to retrieve envelope information, be sure to do the actual receive of the message, otherwise the message will remain queued to the port. Also, consider potential traps and escapes and do not allow your handler to be exited in this fashion. If a handler does escape, it will be caught by AIF ports code and will NOT be propagated out to the user.

Finally, a handler may never call `AIFPORTCLOSE` to close an asynchronous port. This will result in unpredictable behavior, and possible system failure. Handlers that are written in C, may not use the longjump function and must have a return-type of void. Handlers execute at ring level 2 or privileged mode. Calls to `GETUSERMODE` are not allowed inside the handler, this will cause an IMEM protection trap, which results in a process abort.

Special Considerations

The asynchronous receipt of incoming messages has been implemented through Process Interrupts. A process interrupt is generated to signal the arrival of a message on an asynchronous port. The process interrupt will "interrupt" the creator process transferring control to the user supplied interrupt handler. As with other types of process interrupts (eg., Break, Cntrl Y), control will not be transferred to the user's handler until the creator process is in the appropriate state. For asynchronous port interrupts, the process interrupt will be postponed if the creator is critical, in system code, or executing at ring level 0 or 1. This is done to protect critical system operations from being interrupted by the user application.

A waited receive against an asynchronous port takes precedence over notification by a process interrupt. Therefore, a process which blocks, waiting for a message from its asynchronous port has effectively disabled process interrupt notification. Several possible uses of asynchronous ports are described to illustrate the systems behavior.

Example 1

A process opens an asynchronous port with port interrupt handling enabled. The handler does a nowait receive to get the message and it does not access any global data.

When a send is issued against the port the message is queued by the send request. Messages sent to a process NOT blocked on an asynchronous port receive, will result in a process interrupt. When the receiving process is in the appropriate state (ring level 2 or 3) the user handler will be invoked. It is possible for the user interrupt handler to nest multiple levels deep if additional process interrupts occur. In this case, the user should do a single receive against the port for each call of the handler. The user should also handle error messages appropriately.

However, if the receiver chooses to explicitly wait for the arrival of a message on an asynchronous port, when the send is issued, the dispatcher is notified to unblock the process. When the receiver process is awoken, it will complete the receive operation on a single message. It is possible for multiple messages to queue while the process is blocked. If a user waits on an asynchronous port in this fashion, they are responsible for checking for multiple messages once the receive completes.

Example 2

The user opens an asynchronous port. `AIFPORTINT` is called to disable port interrupt handling around critical areas in the user code. Also `AIFPORTINT` is used in the handler to disable interrupt handling after entry, and later re-enabled before exiting the handling routine.

Again, the message is queued to the port when the send request is issued. If the receiver is not currently waiting on the port, the process interrupt will occur. Further process interrupts will nest until the user calls `AIFPORTINT` inside their handler. Once interrupt handling is disable, additional messages which arrive will cause a pending count to be incremented and the handler invokation will be delayed. When `AIFPORTINT` is called to re-enable interrupt handling, the user handler will be called once for each time the pending count was incremented. This is repeated until the pending interrupts have been serviced.

If the user is sitting in a waited receive, multiple messages can be sent, and the receiver will unblock. The receive will return the first message, but the user is responsible for clearing the port. If the user calls `AIFPORTINT` around the waited receive, it is possible the user can clear messages from the port, and when `AIFPORTINT` is called to enable interrupt handling, the handler could get invoked for messages which have already been read. Therefore, the handler should be coded to handle the case where no messages exist on the port.

AIFPORTOPEN Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with `AIFPORTOPEN`.

Table 3-23. AIFPORTOPEN Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
11201	**Create option (I32); Release 3.0** Passes a port creation option. Values and their meanings are as follows: 1 Create a new port if the named port does not exist; otherwise, open the existing port. 2 Create a new port and open it. Return an error if the port already exists. 3 Open an existing port. Return an error if it does not exist. Default: 1
11202	**Maximum message size (I32); Release 3.0** Passes the maximum message size, in bytes, to allow through this port. An error is returned if an attempt is made to send a message larger than this value. Default: 256 bytes Maximum: 8144
11203	**Normal message size (I32); Release 3.0** Passes the normal message size, in bytes. When this number is multiplied by the maximum number of normal messages, the result must be greater than or equal to the maximum message size. Default: 64 bytes
11204	**Maximum # of normal messages (I32); Release 3.0** Passes the maximum number of normal-sized messages (refer to item number 11203) to allow in this port. When this number is multiplied by the normal message size, the result must be greater than or equal to the maximum message size (refer to item number 11202). The absolute maximum number of normal messages is 32,767; however, this value may be smaller based on the following calculation: Maximum # of normal messages = 1,048,256 words / (maximum message size rounded up in words + 16 words) NOTE: The system allocates message pools based on the message size and number of normal messages when the port is created. These resources should be allocated sparingly since they are allocated out of system space. Shortages of system space can result in a system failure. Default: 32

Table 3-23. AIFPORTOPEN Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
11205	**Make permanent (B); Release 3.0** Passes a value specifying the final disposition of the port (whether permanent or removed) after the last process has done a close on it. If the port is to remain after the last process has done a close on it, a value of true must be passed with this parameter for all opens of the port. The last open of the port establishes the final permanence of the port. If the last opener passes this option with a true value, the port is permanent. If the last opener does not specify this option, or specifies it and passes a false value, the port is removed after the last process closes it. To remove a permanent port from the system, all that is required is for a process to open the port without specifying this parameter, or specifying this parameter as false; the port is then destroyed when the last accessor closes the port. An asynchronous port is always temporary. When an asynchronous port is opened, an error is returned if this option is specified as permanent. Default: FALSE (port is temporary)
11206	**Handler address (@32); Release 4.0** Passes the handler address for an asynchronous port. The address of the user defined handler can be acquired by calling the intrinsic `HPGETPROCLABEL`. See the operational notes for handler requirements. This item can only be specified when the port is first created (refer to item 11201). Any attempt to pass this item to an already open port results in an error. An asynchronous port may not be permanent. (Refer to item 11205.) Default: nil
11207	**Interrupt handler state (B); Release 4.0** Passes a Boolean that enables or disables the port interrupt handler at creation time. A value of TRUE means that the interrupt handler is enabled upon return from `AIFPORTOPEN`, while FALSE means that the interrupt handler is disabled and a call to `AIFPORTINT` is required to enable it. Default: FALSE

AIFPORTRECEIVE

Receives a message through a previously opened port.

Syntax

```
                        REC          I32         CA              I32
AIFPORTRECEIVE (overall_status, port_id, message_buffer, message_length,
                       I32          I32          I32A          @64A
                envelope_code, message_id, itemnum_array, item_array,
                        RECA
                itemstatus_array);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

Passes a port ID, returned from a successful call to `AIFPORTOPEN`. This parameter specifies from which port to receive the message. If you specify a ***port_id*** of zero, the next message received from any port previously opened by the calling process is returned, and the ***port_id*** of the port from which the message was taken is returned in this parameter.

message_buffer **character array (required)**

Returns the message. The buffer passed must be large enough to hold the message, or the message is truncated.

Array type: `message_buffer_type` (Refer to appendix B.)

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

Passes the length, in bytes, of ***message_buffer***. (If the message returned is longer than this length, the message is truncated.)

Returns the actual length of the message returned in ***message_buffer*** if the actual length is shorter than the value passed.

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

Returns an integer code associated with the envelope portion of the message. The use of this value is application dependent (for example, it can be used to identify the type of message being received without accessing the actual message buffer).

Default: nil

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

Returns the message ID assigned to this message when it was sent by `AIFPORTSEND`.

Default: nil

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the *item_array* parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the *itemnum_array* parameter, each of which is a `globalanyptr` that points to the appropriate type needed by each particular item number. The value used for each option is taken from, or returned to, the location pointed to by the `globalanyptr` in this array. When this parameter is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr`

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with specific items, an error status is placed in the corresponding element of this array for each item with an error. The overall status parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and *item_array* parameters.

A nonzero value indicates an error, but a valid option does not set the value to zero, so this array should be initialized to all zeros before making the call.

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

Default: nil

Operation Notes

Several options are included with `AIFPORTRECEIVE` to allow increased control over the delivery of each message. Some of the most significant options are the ability to wait for the message to be delivered and the ability to time out if the message is not received within a given number of seconds.

For asynchronous ports it is very important for the user to properly manage the receipt of messages on the port. When a port has its interrupt handler enabled, it is possible for multiple messages to arrive, causing nested interrupts. Inside a handler, the user should receive a single message, even though muliple messages could exist on the port. Clearing a message does not prevent the handler from being invoked.

There is an exception, however: when AIFPORTINT was used to disable port interrupt handling, newly arriving messages do not cause the handler to be invoked. Interrupt handling is delayed, and a pending count is incremented. After AIFPORTINT is used inside the handler, the first receive should pick up the message that caused the handler to be called. The user can then issue another receive with item 11007 to get a message with a pending count. When there are messages with a pending count, the receive succeeds and the message is returned. The receive can be called repeatedly with item 11007 until an error is returned indicating that there are no more messages with pending interrupts. Each AIFPORTRECEIVE with item 11007 will decrement the pending count. When the pending count is 0, delayed calls of the interrupt handler do not occur.

AIFPORTRECEIVE Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with `AIFPORTRECEIVE`.

Table 3-24. AIFPORTRECEIVE Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
11001	**Priority mask (I32); Release 3.0** Passes a priority bit mask that determines which messages are received. A message can be sent at any of 32 possible priorities. If this option is specified, only messages that come in with the indicated priorities are received. This parameter is a bit mask with each bit position, from left to right, indicating the corresponding priority, 0 to 31, that should be received. For example, if the third bit is on with all other bits off, only messages that have a priority of 2 are received. Remember that the leftmost bit is bit zero, and the bits are numbered left to right.
11002	**Time out seconds (I32); Release 3.0** Passes a value that sets a time out value in seconds. If the message is not received within the number of seconds specified, `AIFPORTRECEIVE` fails, and a status indicating that the timeout has expired is returned. Following are valid values and their meanings: -1 Don't wait. Specifying a timeout of -1 signals this receive to be a nowait receive. 0 Wait indefinitely to receive the message. >0 Wait the specified number of seconds for a receiver to get the message, then return an error status. Default: 0 (wait indefinitely)
11003	**Message return (B); Release 3.0** Passes a value that allows the retrieval of pieces of information from the envelope without getting the message portion of the package. Following are the possible values and their meanings : TRUE Return the message to the specified message buffer. If the message is longer than the length of the buffer, it is truncated. There is no indication returned that the message has been truncated. FALSE Do not return the message. When this option is used, the next `AIFPORTRECEIVE` call to the same port (with this option set to true) returns the message. Other parameters in the `AIFPORTRECEIVE` call (for example, *envelope_code*, *message_id*, ***message_length***, and ***port_id***) are returned with information that may be useful at a later time. Both *envelope_code* and ***message_length***, in particular, can be used to determine the application-defined type of message, and if the available buffer space is enough before the message is received and truncated because the buffer is not big enough. Default: TRUE

Table 3-24. AIFPORTRECEIVE Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
11004	**Sender PID (I32); Release 3.0** Returns the sender's process ID (PID).
11005	**Sender PIN (I32); Release 3.0** Returns the sender's process identification number (PIN). The PIN is a 16-bit value, but is returned as an 32-bit integer.
11006	**Actual priority (I32); Release 3.0** Returns the priority of the message. Unless messages are being received only from a specific priority (See item 11001 priority mask), there is no way to tell the priority of the message just received unless this option is used.
11007	**Message with pending interrupt (B); Release 4.0** Passes a Boolean that when set to TRUE, indicates a request to receive a message that has a pending interrupt. Messages with pending interrupts are caused by calls to the `AIFPORTINT` routine, which disables interrupt handling on a specified port. Messages that arrive on a port after disabling interrupt handling cause a pending interrupt count to be incremented for each message that has arrived provided the receiver is not waiting on the port. When the `AIFPORTINT` routine is called to enable interrupts, the user handler is called once for each pending interrupt. When this option is used on an `AIFPORTRECEIVE` with a value of TRUE and there is a pending interrupt count greater than 0, the message is received and the pending interrupt count is decremented by one. When there are no messages with pending interrupts, an error is returned. (Refer to appendix A.) A port ID of zero cannot be used with this option. Default: FALSE

AIFPORTSEND

Sends a message to another process through a previously opened port.

Syntax

```
                    REC         I32          CA              I32
AIFPORTSEND (overall_status, port_id, message_buffer, message_length,
                    I32         I32          I32A            @64A
             envelope_code, message_id, itemnum_array, item_array,
                    RECA
             itemstatus_array);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

Passes a port ID, returned from a successful call to `AIFPORTOPEN`. This parameter specifies the port that is to receive the message.

message_buffer **character array (required)**

Passes the actual message to send through the specified port.

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

Passes the length, in bytes, of the message buffer to send through the specified port.

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

Passes an integer code associated with the envelope portion of the message. The use of this value is application dependent; for example, it can be used to identify the type of message being sent, so the receiving process can identify the message type without accessing the actual message buffer. If this parameter is not supplied, *envelope_code* is defaulted to zero.

Default: 0

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

A code returned by `AIFPORTSEND` to identify this particular message.

Default: nil

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

This is an array of integers, terminated by an element containing the value zero, used to define the corresponding option given in the *item_array* parameter. If this optional parameter is specified, the *item_array* parameter and the *itemstatus_array* parameter must both be supplied.

Default: nil

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

An array with the same number of elements as the *itemnum_array* parameter, each of which is a `globalanyptr` that points to the appropriate type needed by each particular item number. The value used for each option is taken from, or returned to, the location pointed to by the `globalanyptr` in this array. When this parameter is supplied, the *itemnum_array* parameter and the *itemstatus_array* parameter must both be supplied.

Array type: `globalanyptr`

Default: nil

itemstatus_array **record array by reference (optional)**

If problems are detected with specific items, an error status is placed in the corresponding element of this array for each item with an error. The overall status parameter indicates whether any individual items contained errors, and the element of the last detected error. This array must contain as many elements as are contained in the *itemnum_array* and *item_array* parameters.

A nonzero value indicates an error, but a valid option does not set the value to zero, so this array should be initialized to all zeros before making the call.

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

Default: nil

Operation Notes

Several optional items allow `AIFPORTSEND` increased control over the delivery of each message. Some of the most significant options are the ability to wait for the message to be received, and the ability to time out if the message is not received within a given number of seconds.

It is possible to send a message to a port that was not explicitly opened by the caller. The `AIFPORTSEND` must use item 11101 with a `nowait` value of -1. A `nowait` send queues the message to the port and returns immediately to the caller and does not wait for a receive to be issued against the port.

AIFPORTSEND Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with `AIFPORTSEND`.

Table 3-25. AIFPORTSEND Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
11101	**Time out seconds (I32); Release 3.0** Passes a value that sets a timeout in seconds. If the message is not received within the number of seconds specified, `AIFPORTSEND` fails, and a status indicating that the timeout has expired is returned. Following are valid values and their meanings: -1 Don't wait. Specifying a timeout of -1 signals this send to be a `nowait` send. Control is returned to the caller as soon as the message has been placed in the specified port. 0 Wait indefinitely for a receiver to get the message. >0 Wait the specified number of seconds for a receiver to get the message, then destroy the message (no process will receive it) and return an error status. Default: 0 (wait indefinitely)
11102	**Priority (I32); Release 3.0** Passes the priority to use in sending this message. The possible values range from 0 to 31, with 0 being the highest priority. If priorities are used, the messages are no longer guaranteed to be received in the same order in which they were sent. Default: 0
11103	**Connectionless send (B); Release 4.0** Passes a boolean that indicates that a message may be sent to a port that has not been previously opened for send access. This item does not allow item 11101 to be specified with a value >= 0. This means that a connectionless send may only be done as a "no wait" send. If item 11101 is specified with an illegal value, an error is returned. Default: FALSE

AIFPROCGET

Returns process information.

Syntax

```
                          REC        I32A
AIFPROCGET(overall_status, itemnum_array,
                         @64A         RECA
                    item_array, itemstatus_array,
                    I32    REC     I32
                    PIN, PID, user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the information to be returned to a data structure pointed to in the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

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

Passes the process identification number (PIN) of the process for which information is desired.

Default 0

PID **Record by value (optional)**

Passes the process identifier (PID) of the process for which information is desired.

Record type: longint_type

Default 0

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

`AIFPROCGET` accepts either of the following as an input key:

- A process identification number (PIN), that identifies a process immediately and provides faster access than using the PID. However, PINs are not unique throughout the life of a system. Thus, there is a chance that the specified PIN is associated with a different process than expected.
- A process identifier (PID), that uniquely identifies a process throughout the life of a system. Using a PID to access process information is almost as fast as using a PIN.

If neither *PIN* or *PID* are provided, the default is the PIN of the calling process.

AIFPROCPUT

Modifies process information

Syntax

```
                          REC          I32A
AIFPROCPUT(overall_status, itemnum_array,
                         @64A           RECA
                    item_array, itemstatus_array,
                    I32    REC    I32
                    PIN, PID, user_id,
                      I32A              @64A             RECA
                    ver_item_nums, ver_items, ver_item_statuses )
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure containg new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

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

Passes the process identification number (PIN) of the process whose information is to be modified.

Default 0

PID **Record by value (optional)**

Passes the process identifier (PID) of the process whose information information is to be modified.

Record type: longint_type

Default 0

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in ver_items. if n items are being verified, element n+1 must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in ver_item_nums.

Array type: globalanyptr

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in ver_items. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

Operation Notes

`AIFPROCPUT` accepts either of the following as an input key:

- A process identification number (PIN) that identifies a process immediately and provides faster access than using the PID. However, PINs are not unique throughout the life of a system. Thus, there is a chance that the specified PIN is associated with a different process than expected.
- A process identifier (PID) that uniquely identifies a process throughout the life of a system. Using a PID to access process information is almost as fast as using a PIN.

If neither *PIN* or *PID* are provided, the default is the PIN of the calling process.

The process whose information is being modified must be of type user, son, or CI. If it is anything else, `AIFPROCPUT` terminates with an error condition.

If both the PIN and PID are provided, the values are checked against each other. If the process identifiers do not match, `AIFPROCPUT` terminates with an error condition.

AIFPROCGET/PUT Items

The following two tables provide summary and detailed descriptions of the items associated with process information.

Item Summary

The following table summarizes the item numbers associated with process information. For more detailed information about these item numbers, refer to the table of process information item descriptions.

Table 3-26. Process Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
2001	Longint_type	PID	N	Y			
2002	I32	PIN	N	Y			
2003	Longint_type	Parent PID	N	Y			
2004	I32	Parent PIN	N	Y			
2005	Longint_type	Sibling PID	N	Y			
2006	I32	Sibling PIN	N	Y			
2007	Longint_type	Child PID	N	Y			
2008	I32	Child PIN	N	Y			
2009	Longint_type	JSmain PID	N	Y			
2010	I32	JSmain PIN	N	Y			
2011	Longint_type	Last child PID	N	Y			
2012	I32	Last child PIN	N	Y			
2013	Longint_type	Creator PID	N	Y			
2014	I32	Creator PIN	N	Y			
2015	I32	Job/session number	N	Y			
2016	I32	Scheduling state	N	Y			
2017	I32	Scheduling queue	Y	Y	0	4	-2008
2018	B	Degradable priority?	Y	Y			
2019	I32	Priority	Y	Y	0	32767	-2009
2020	U32	Reasons for boost	N	Y			
2021	I32	Post boost priority	N	Y			
2022	I32	Process state	N	Y			
2023	Longint_type	Waiting time (ticks)	N	Y			
2024	Longint_type	Waiting time (msecs)	N	Y			
2025	I32	Waiting reason	N	Y			
2026	I32	NM error queue head	Y	Y	0	16	-2007
2027	I32	NM error queue tail	Y	Y	0	16	-2007
2028	B	Lost NM error entries?	Y	Y			
2029	I32	Number of NM errors	Y	Y	0	16	-2007
2030	I32A	List of NM errors	Y	N			

Table 3-26. Process Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
2031	I32	I/O count	N	Y			
2032	I32	CM I/O count	N	Y			
2033	U32	Process type	N	Y			
2034	Filename_type	Program name	N	Y			
2035	I32	Program file number	N	Y			
2036	A64	Entry address	N	Y			
2037	B	CM mode initially?	N	Y			
2038	B	Info string passed?	N	Y			
2039	CA256	Info string	N	Y			
2040	I32	Parm	N	Y			
2041	I32	SR5 space ID	N	Y			
2042	A64	XRT area base	N	Y			
2043	A64	XRT area limit	N	Y			
2044	A64	CM area base	N	Y			
2045	A64	CM area limit	N	Y			
2046	A64	NM stack base	N	Y			
2047	A64	NM stack limit	N	Y			
2048	A64	Heap area base	N	Y			
2049	A64	Heap area limit	N	Y			
2050	A64	PCBX address	N	Y			
2051	B	Split stack mode?	N	Y			
2052	I32	DB DST number	N	Y			
2053	I32	CM stack DST number	N	Y			
2054	A64	DB pointer	N	Y			
2055	A64	DL pointer	N	Y			
2056	I32	Initial DL	N	Y			
2057	I32	Initial Q	N	Y			
2058	I32	Number of XDS	N	Y			
2059	Dstsrec_type	List of XDS	N	N			
2060	I32	LSTT DST number	N	Y			
2061	A64	LSTT address	N	Y			
2062	I32	Number of open files	N	Y			
2063	I32rec_type	Open file numbers	N	N			
2064	Fnamerec_type	Open file names	N	N			
2065	Ufidrec_type	Open file UFIDs	N	N			
2066	I64rec_type	List of child PIDs	N	N			

Table 3-26. Process Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
2067	A64	PCB pointer	N	Y			
2068	I32	Maximum allowed short mapped space	N	Y			
2069	I32	Used short-mapped space	N	Y			
2070	I32	General resource capabilities	Y	Y			
2071	I32	System code depth	N	Y			
2072	I32	Critical code depth	N	Y			
2073	I32	Number of CM errors	Y	Y	0	6	-2006
2074	I32A	List of CM errors	Y	N	-32768	32767	-2005
2075	I32	Last `FOPEN` error	Y	Y	0	32767	-2007
2076	I32	Last `KOPEN` error	Y	Y	0	255	-2007
2077	B	CM aritraps enabled?	N	Y			
2078	I32	CM aritrap handler plabel	N	Y			
2079	I32	NM aritrap mask	N	Y			
2080	A64	NM aritrap handler address	N	Y			
2081	I32	CM libtrap handler plabel	N	Y			
2082	A64	NM libtrap handler address	N	Y			
2083	I32	CM systrap handler plabel	N	Y			
2084	I32	Privileged level of NM systrap	N	Y			
2085	A64	NM systrap handler address	N	Y			
2086	I32	UNSAT handler address	N	Y			
2087	CA32	UNSAT handler name	N	Y			
2088	B	Dump armed?	Y	Y			
2089	CA256	Debug commands	Y	Y			
2090	B	Debug armed?	Y	Y			
2091	Longint_type	CPU time (ticks)	N	Y			
2092	Longint_type	CPU time (msecs)	N	Y			
2093	B	SIR holder?	N	Y			
2094	I32	JS Key	N	Y			
2095	I32	User and file access capabilities	Y	Y			
2096	Longint_type	Time process on Ready Queue	N	Y			
2105	@64	NM stack maximum SP	N	Y			
2106	B	Execution Mode	N	Y			
2107	I32	CM Maxdata	N	Y			
2108	I32	CM S	N	Y			
2109	I32	JDT DST	N	Y			

Table 3-26. Process Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
2110	CA16	Job name	Y	Y			
2111	CA16	User name	N	Y			
2112	CA16	Group name	N	Y			
2113	CA16	Account name	N	Y			
2114	I32	Maximum account job priority	N	Y			
2115	I32	Account security	Y	Y			
2116	I32	Group security	Y	Y			
2117	CA16	Home group	N	Y			
2118	I32	Account local attributes	Y	Y			
2119	I32	User capabilities	Y	Y			
2120	I32	General resource capabilities	Y	Y			
2121	BA96	Allow mask	Y	Y			
2122	@64	Pathnames of open files	N	N			
2123	RECA	Path Identifiers of open files	N	Y			
2125	B	Fork Process	N	Y			
2126	I32	UID	Y	Y			
2127	I32	EUID	Y	Y			
2128	I32	GID	Y	Y			
2129	I32	EGID	Y	Y			
2130	U32	CMASK	Y	Y			
2131	REC	Program pathname	N	Y			
2132	B	Break Request Done	N	Y			
2133	I32	Break Request Cancel	N	Y			
2134	I32	Break Request Pending	N	Y			
2135	REC	List of sibling PIDs	N	Y			
2136	REC	List of parent PIDs	N	Y			
2142	B	Interactive?	Y	Y			
2143	B	Environment Nil	N	Y			
2144	CA256	Workgroup name	Y	Y			
2145	B	Artificial workgroup member	N	Y			
2146	B	Return natural workgroup	Y	N			
2147	I32	Execution state	N	Y			
2148	I32	Fixed priority	Y	N	0	32767	

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with process information.

Table 3-27. Process Information Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Description
2001	**PID (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the process. Record type: `longint_type` (Refer to appendix B.)
2002	**PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the process.
2003	**Parent PID (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the parent process. Record type: `longint_type` (Refer to appendix B.)
2004	**Parent PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the parent process.
2005	**Sibling PID (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the sibling process (the next sibling in chronological order). All the children of a process are linked together in one direction. The head of the list is always at *parent.child*. A value of 0 indicates the end of the sibling list. Record type: `longint_type` (Refer to appendix B.)
2006	**Sibling PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the sibling process (the next sibling in chronological order). All the children of a process are linked together in one direction. The head of the list is always at *parent.child*. A value of 0 indicates the end of the sibling list.
2007	**Child PID (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the first child process created by the specified process. A PID of 0 indicates that no child process exists. Record type: `longint_type` (Refer to appendix B.)
2008	**Child PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the first child process created by the specified process. A PIN of 0 indicates that no child process exists.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2009	**JSmain PID (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the JSmain process of the tree to which this process belongs. For system processes, a 0 is returned. For Jsmains in use, its own PID is the also JSmain PID. Record type: `longint_type` (Refer to appendix B.)
2010	**JSmain PIN (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the JSmain process of the tree to which this process belongs. For processes of type system, detach, and task, a 0 is returned.
2011	**PID of the last child process (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the last child created by this process. Because the last child created may no longer exist, the PID should be used carefully. A 0 is returned if no child process was ever created. Record type: `longint_type` (Refer to appendix B.)
2012	**PIN of the last child process (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the last child created by this process. Because the last child created may no longer exist, the PIN should be used carefully. A 0 is returned if no child process was ever created.
2013	**PID of creator (REC) Put: No; Verify: Yes; Release 3.0** Returns the PID of the creator process, usually the parent process. However, some system processes are adopted to another parent after creation. These processes have a different creator. Record type: `longint_type` (Refer to appendix B.)
2014	**PIN of creator (I32) Put: No; Verify: Yes; Release 3.0** Returns the PIN of the creator process, usually the parent process. However, some system processes are adopted to another parent after creation. These processes have a different creator.
2015	**Job/session number (I32) Put: No; Verify: Yes; Release 3.0** Returns the job/session number of the job/session domain to which the process belongs. This number is valid for processes of the type user and son. For all other processes, a 0 is returned. It also returns a 0 for some user processes like VTSERVER and NFT. A negative number indicates a job; a positive number indicates a session. The job/session number for this job or session in the following format: Bits (0:2) Job or session? (1 = Session, 2 = Job) Bits (2:14) Number Bits (16:16) Extension

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2016	**Scheduling state (I32) Put: No; Verify: Yes; Release 3.0** Returns the state of the process, as viewed by the dispatcher. It is the first item that should be interrogated to ascertain a process's state. Values and their meanings are as follows: 0 Executing (only for Calling Process) 1 Ready 2 Short wait 3 Long wait 4 Null Processes in the ready queue are linked together in the order of priority. A short wait is basically a wait for disk I/O, and the dispatcher expects the process to be ready in a short while. See the item 2025 "Reason for waiting" for further information. The null state is seen only for processes that are dead or in the process of dying.
2017	**Scheduling queue (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the scheduling queue that this process belongs to. Values and their meaning are as follows: 0 AS Queue 1 BS Queue 2 CS Queue 3 DS Queue 4 ES Queue Modifying this information causes the process to be placed in the specified queue, with the priority being the base of the new queue.
2018	**Degradable priority? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies whether the process undergoes priority decay. True indicates that the process will undergo the normal priority decay from base to limit to base of the scheduled class it is in (CS, DS, ES), while it is in a circular queue. False causes the priority to remain fixed at the current priority. Classes are obtaining through `AIFSCGET`. This item makes sense only for processes in the circular classes (CS, DS, ES) since the linear queue processes do not undergo priority decay. Also, a process that has gone through decayable boosting is always subject to priority drop. This also means that it contributes towards the system CS-SAQ. Scheduling classes can be obtained through `AIFSCGET`.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description

2019	**Priority (I32) Put: Yes; Verify: Yes; Release 3.0**
	
	Returns or modifies the priority of the process. **This is an MPE/iX priority.** It is very transient for user processes. For processes whose priority is not fixed, this value should be interpreted as the priority at which the process was last dispatched. For nonconstant priority processes, it is not used in determining the priority at which it will be dispatched next.
	
	A valid MPE/iX priority is in the range 0..32767. The new priority should be in the range of priorities specified by the base and the limit of the current scheduling class of the process. This priority can be mapped to MPE V by the following formula:
	
	`MPEVPri = (32767 - MPEXLPri) DIV 128  (All formula values are decimal)`
	
	`ex.  B149  = (32767 - 13695)  DIV 128`

2020	**Reasons for boost (U32) Put: No; Verify: Yes; Release 3.0** Returns the reasons for the priority boost. The bits and their meanings are as follows: Bit (0:1) The process is experiencing a non-decayable boost because the process owns a priority semaphore or resource for which there is a contention. Bit (1:1) The process is experiencing a non-decayable boost because the process owns a SIR for which there is a contention. Bit (2:1) The process is experiencing a non-decayable boost because the process has a long-running system transaction. Bit (3:1) The process is experiencing a non-decayable boost to ensure prompt handling of a system or subsystem break event. Bit (4:1) The process is experiencing a decayable boost because the process owns a priority semaphore or resource for which there is contention. Bit (5:1) The process is experiencing a non-decayable boost because the process is currently deemed unpreemptable and has blocked. Bit (6:1) The process is experiencing a non-decayable boost because the process is hosting a IPC server for which there is contention. Bit (7:1) The process is experiencing a decayable boost because the process has a long-running user transaction. Bit (8:1) The process is experiencing a non-decayable boost because the process owns a priority semaphore port for which there is contention. Bit (9:1) The process, a serial printer server, is experiencing a decayable boost to force priority oscillation. If no bit is turned on, the process priority has not been boosted.
2021	**Post boost priority (I32) Put: No; Verify: Yes; Release 3.0** Returns the post boost priority of the process. This is an MPE/iX priority with a range of 0..32767. This is the priority that will be in effect after the process has unboosted from a new priority to which it was boosted for some purpose. The process will be reassigned this priority as soon as possible. A 0 is returned if the process is not currently boosted.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2022	**Process state (I32) Put: No; Verify: Yes; Release 3.0** Returns the state of the process from the viewpoint of process management. In general, it should be alive for most processes. The other states are generally very transient. The data returned is valid mainly for the alive case. Values and their meanings are as follows: 0 Unknown 1 Dying 2 Dead 3 Alive 4 Initiate 5 Unborn
2023	**Time in ticks, when it began waiting (REC) Put: No; Verify: Yes; Release 3.0** Returns the time, in ticks, since 1970 when the process entered the wait state. This field is only updated when the Measurement Interface is turned on. This time is processor dependent. To obtain processor-independent time, use the item 2024 for time in milliseconds. (This item provides faster access to time than item 2024.) Record type: `longint_type` (Refer to appendix B.)
2024	**Time in milliseconds, when it began waiting (REC) Put: No; Verify: Yes; Release 3.0** Returns the time, in milliseconds, since 1970 when the process entered the wait state. This field is only updated when the Measurement Interface is turned on. This time is processor independent. Record type: `longint_type` (Refer to appendix B.)

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2025	**Reason for waiting (I32) Put: No; Verify: Yes; Release 3.0** Returns reasons that a process is not currently executing. 0 nm code page fault 1 nm stack page fault 2 nm transient page fault 3 file page fault 4 cm code page fault 5 cm stack page fault 6 cm transient page fault {The page fault reasons 0..6 are returned when the Measurement Interface is turned on. Otherwise, disc io wait (9) will be returned.} 7 terminal read wait 8 terminal write wait 9 disc io wait 10 other io wait 11 ipc trans complete 12 sir wait 13 rin wait 14 memory manager prefetch 15 quantum expiration 16 timer wait 17 parent wait 18 control block wait 19 child wait 20 data comm wait 21 rit wait 22 disp work 23 port wait { the following are subevents of port wait } 24 mail wait 25 junk wait 26 message wait 27 impede 28 break wait 29 wait queue 30 memory management wait 31 port blocked make present 32 file blocked 33 file unblocked 34 storage management 35 user to debug message 36 io configuration wait 37 pfp reply wait 38 db monitor wait 39 fill disc wait 40 hlio wait

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2025	**Reason for waiting (continued from previous page); Release 3.0** 41 file system terminal io wait 42 memory manager post wait 43 signal timer wait 44 preemption 45 disc io preemption 46 priority preemption 47 sql lock wait 48 sql latch level 1 wait 49 sql latch level 2 wait 50 sql latch level 3 wait 51 sql latch level 4 wait 52 sql latch level 5 wait 53 sql latch level 6 wait 54 sql latch level 7 wait 55 sql latch level 8 wait 56 sql latch level 9 wait 57 sql latch level 10 wait 58 sql latch level 11 wait 59 sql latch level 12 wait 60 sql latch level 13 wait 61 sql latch level 14 wait 62 sql latch level 15 wait 63 sql latch level 16 wait 64 sql latch level 17 wait 65 sql latch level 18 wait 66 sql latch level 19 wait 67 sql latch level 20 wait 68 sql latch level 21 wait 69 sql latch level 22 wait 70 sql latch level 23 wait 71 sql latch level 24 wait 72 sql latch level 25 wait 73 sql latch level 26 wait 74 sql latch level 27 wait 75 sql latch level 28 wait 76 sql latch level 29 wait 77 sql latch level 30 wait 78 sql latch level 31 wait 79 sql latch level 32 wait

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2025	**Reason for waiting (continued from previous page); Release 3.0** 80 sql buffer wait 81 long pause_wait 82 memory manager freeze and other 83 release 84 deferred preempt 85 memory manager pseudo ioread 86 memory manager pseudo iowrite 87 other wait 100 dispatcher not blocked 101 dead process
2026	**Last NM error entry number (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the last NM error entry number, a value in the range 0..16. The NM error object is a circular queue of 16 elements. Upon entry into an NM intrinsic, the NM intrinsic error object is flushed out. The last error returned here is an index into the error object, to the rear of the error queue. It is reset to 0 upon entry into an intrinsic.
2027	**First NM error entry number (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the first error entry number, a value in the range 0..16. The NM error circular queue may wrap around in case of too many errors. This item returns an index to the first valid error message, that is, the new front of the circular queue. It is reset to 0 upon entry into an intrinsic.
2028	**Any NM errors lost? (B) Put: Yes; Verify: Yes; Release 3.0** True if the error queue has wrapped around, causing errors to be lost. It is reset to false upon entry into an intrinsic.
2029	**Total number of NM errors (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of valid errors recorded in the error queue, a value in the range 0..16. It is reset to 0 upon entry into an intrinsic.
2030	**NM intrinsic errors (I32A) Put: Yes; Verify: No; Release 3.0** Returns an array of all the errors dumped onto the stack by the last call to an NM intrinsic. The errors are all MPE/iX statuses that can be investigated through normal error mechanisms. The range of indices holding valid errors is determined by the above items. The maximum number of errors is 16. The user should pass an area of appropriate size. The first word of the buffer is expected to hold the size, in words, of the rest of the buffer area. The first word, on return, specifies the number of errors returned. The *itemstatus_array* should be checked to determine whether information was truncated.
2031	**I/Os outstanding (I32) Put: No; Verify: Yes; Release 3.0** Returns the total number of I/Os outstanding for this process.
2032	**CM I/Os outstanding (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of CM I/Os outstanding for this process.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2033	**Process type (U32) Put: No; Verify: Yes; Release 3.0** Returns the type of the process. Values and their meanings are as follows: 0 User (any process created by a user) 1 Son (process created by CI to run user programs) 2 Main (CI process) 3 Task (not in use) 4 System (some integral processes) 5 Detach (not connected to the PROGEN tree) 6 UCOP (JSmain) 7 Unknown (uninitialized processes)
2034	**Program name (REC) Put: No; Verify: Yes; Release 3.0** Returns the fully qualified MPE syntax name of the program file. It is of type `filename_type`, with the file, group, and account names each left-justified and padded with blanks. Note that this item should be used only for names that can be expressed using MPE syntax. Item 2131 should be used for HFS syntax or MPE syntax program files which are represented using a HFS pathname. If you select this item for a file that cannot be represented using MPE syntax, then blanks are returned and a warning is issued in itemstatus_array. Record type: `filename_type` (Refer to appendix B.)
2035	**Program file number (I32) Put: No; Verify: Yes; Release 3.0** Returns the `HPFOPEN` process local file number for the program file.
2036	**Entry pointer (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of the entry point for the program.
2037	**CM mode initially? (B) Put: No; Verify: Yes; Release 3.0** Returns true if the image was loaded from a CM program and false if it was loaded from an NM program.
2038	**Info string passed? (B) Put: No; Verify: Yes; Release 3.0** Returns true if an info string was passed when the program was loaded.
2039	**Info string (CA256) Put: No; Verify: Yes; Release 3.0** Returns the character array (of maximum 256 characters) that was passed when the program was loaded. It is valid only if item 2038 "Info string passed?" is true. It is left-justified and padded with blanks.
2040	**Parm (I32) Put: No; Verify: Yes; Release 3.0** Returns the parm specified (if any) when the program was loaded. By default it is always 0.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2041	**Space ID of the stack (I32) Put: No; Verify: Yes; Release 3.0** Returns the space ID for SR5. This space consists of, among other things, the NM area, the XRT area, and the CM area.
2042	**XRT area base (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the base of the XRT area.
2043	**XRT area limit (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the limit of the XRT area. This area is used for branching to external routines.
2044	**CM area base (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the base of the CM area.
2045	**CM area limit (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the limit of the CM area. The CM area is configured exactly as in MPE V/E. It takes into account the `MAXDATA` specified in the `RUN` command or at process creation time.
2046	**NM stack base (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the base of NM Stack.
2047	**NM stack limit (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the limit of the NM stack. Within the area between the NM stack base and the stack limit lie the stack area, the heap area, and the global data area. It takes into account the size specified at process creation time or in the `NMSTACK` option in the `RUN` command.
2048	**Heap area base (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the base of the heap area.
2049	**Heap area limit (@64) Put: No; Verify: Yes; Release 3.0** Returns a pointer to the limit of the heap area. The heap grows in the area between the heap base and the heap limit. The heap and the stack grow towards each other in Pascal. It takes into account the size specified at process creation time or in the `NMHEAP` option in the `RUN` command.
2050	**PCBX address (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of PCBX base.
2051	**Split stack mode? (B) Put: No; Verify: Yes; Release 3.0** Returns true if the process is currently in split stack mode. By default it is false, even for NM processes.
2052	**DB DST number (I32) Put: No; Verify: Yes; Release 3.0** Returns the DST number of the segment into which DB is pointing. It may be different from the actual stack DST in split stack mode.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2053	**CM stack DST number (I32) Put: No; Verify: Yes; Release 3.0** Returns the DST number assigned to the CM area in the process local space. It is initialized at process creation time, once and for all. The address of the CM area base, in split stack mode, can be obtained through the CM area base and NM SID items.
2054	**DB (@64) Put: No; Verify: Yes; Release 3.0** Returns an offset within the space ID of the current stack. This is maintained only at the time of a switch. It may be outdated information if the process is in CM.
2055	**DL (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of DL. It points into the CM stack area.
2056	**Initial DL (I32) Put: No; Verify: Yes; Release 3.0** Returns the initial displacement from DL to DB, in halfwords (16-bit words). It is the size specified in the `DL` option of the `RUN` command or at process creation time.
2057	**Initial Q (I32) Put: No; Verify: Yes; Release 3.0** Returns the initial displacement from DB to Q, in halfwords (16-bit words). It takes into account the size specified in the `STACK` option at process creation time or in the `RUN` command.
2058	**Number of extra data segments (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of extra data segments allocated to the process.
2059	**List of extra data segments (REC) Put: No; Verify: No; Release 3.0** Returns an array of the DST numbers and the virtual addresses of the extra data segments allocated to the process. The maximum number of DSTs for a process can be obtained from `AIFSCGET`. You should pass a buffer of appropriate size. The first word of the buffer is expected to hold the size, in words, of the rest of the buffer area. The first word, upon return, specifies the number of DST numbers returned. Check *itemstatus_array* to see if information was truncated. Record type: `dstsrec_type` (Refer to appendix B.)
2060	**LSTT DST number (I32) Put: No; Verify: Yes; Release 3.0** Returns the DST number of the segment holding the logical segment transfer table.
2061	**LSTT address (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of the logical segment transfer table. Using this address is a faster method of accessing the LSTT than the CM way of going through a DST.
2062	**Number of open files (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of files opened for this process. Standard files and all active opens are counted. If a file has been opened twice, it is counted twice.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2063	**File numbers of the open files (REC) Put: No; Verify: No; Release 3.0** Returns an array of the NM file numbers of all the files opened by the process. The maximum number of files can be 1024, including the standard files. Note that this item returns file numbers for both MPE syntax and HFS syntax files. Item 2064 only supports names that can be represented using MPE syntax. Item 2122 is able to represent all file names, including MPE syntax and HFS syntax files. Not all the standard files may be open by default. Hence, some of these may not be returned. The file numbers for the standard files are: 0 `$STDIN` 1 `$STDLIST` 2 `$STDERR` 3 Not used 4 Root directory 5 Account directory 6 Group directory 7 Temporary directory You should pass an area of appropriate size. The first word of the buffer is expected to hold the size, in words, of the rest of the buffer area. The first word, upon return, specifies the number of file numbers returned. Check *itemstatus_array* to see if information was truncated. Record type: `I32rec_type` (Refer to appendix B.)
2064	**MPE names of files (REC) Put: No; Verify: No; Release 3.0** Returns a list of fully qualified file names of the files opened by this process. The file name, group name, and account name are each left-justified and padded with blanks. For device files and standard files, the group and account names will be blanks. Only those names which can be represented by MPE-syntax will be returned. For HFS files, the filename, group, and account fields will be blank and a warning will be returned. Your application can either check for blank names if you wish to continue using this item or you can use item 2122 to retrieve the file pathnames. The maximum number of files that can be opened by a process can be obtained from `AIFSCGET`. You should pass an area of appropriate size. The first word of the buffer is expected to hold the size, in 16-byte records, of the rest of the buffer area. The first word, on return, specifies the number of names returned. Check *itemstatus_array* to see if information was truncated. Record type: `Fnamerec_type` (Refer to appendix B.)

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2065	**UFIDs of files (REC) Put: No; Verify: No; Release 3.0** Returns a list of UFIDs (unique identifiers) for the open files. These can then be used as input to the other AIFs. For device files and standard files, the UFID will be blanks. You should pass an area of appropriate size. The first word of the buffer will be expected to hold the size, in 5-word chunks, of the rest of the buffer area. The first word, on return, specifies the number of UFIDs returned. Check *itemstatus_array* to see if information was truncated. The Pathname Identifier, item 2123, should be specified for HFS files since the UFID alone is not adequate to return a Pathname for an HFS file. Record type: `Ufidrec_type` (Refer to appendix B.)
2066	**Process Tree (REC) Put: No; Verify: No; Release 3.0** Returns entire process tree, PIDs for calling process, PIDs for children, and PIDs for any grandchildren. The children appear in chronological order of birth. The maximum size is a system constant, obtainable from `AIFSCGET`. You should pass a buffer of appropriate size. The first word of the buffer is expected to hold the size, in longwords, of the rest of the buffer area. The first word, upon return, specifies the number of PIDs returned. Check *itemstatus_array* to see if information was truncated. Record type: `I64rec_type` (Refer to appendix B.)
2067	**PCB pointer (@64) Put: No; Verify: Yes; Release 3.0** Returns the PCB pointer.
2068	**Max. short mapped space allowed (I32) Put: No; Verify: Yes; Release 3.0** Returns the maximum amount, in bytes, of short-mapped space allowed.
2069	**Short mapped space used (I32) Put: No; Verify: Yes; Release 3.0** Returns the amount, in bytes, of virtual space used for short-mapped files.
2070	**General resource capabilities (U32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the resources capability mask for the process. This mask contains the resource capabilities for the user associated with the process. Not valid for processes in the wait state. Mask bits and their meanings are as follows: Bits (0:22) Unused Bit (23:1) Batch access Bit (24:1) Interactive access Bit (25:1) Privileged mode Bit (26:2) Unused Bit (28:1) Multiple RINs Bit (29:1) Unused Bit (30:1) Extra data segment Bit (31:1) Process handling
2071	**System code depth (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of nested calls to enter system code.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2072	**Critical code depth (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of nested calls to enter critical code.
2073	**Number of CM intrinsic errors (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of errors during the last call to a CM intrinsic. Values are in the range 0..6. This item is zeroed out upon entry into a CM intrinsic.
2074	**CM intrinsic errors (I32A) Put: Yes; Verify: No; Release 3.0** Returns or modifies the errors dumped onto the stack by the last call to a CM intrinsic. It is flushed out upon entry to any intrinsic. Values are in the range of shortint. The highest index entry is the last error recorded. The format of the error message depends upon the intrinsic called. This array can have a maximum of 6 elements. You should pass an area of appropriate size. The first word of the buffer is expected to hold the size, in words, of the rest of the buffer area. The first word, upon return, specifies the number of error returned. Check *itemstatus_array* to see if information was truncated.
2075	**Last FOPEN error (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the last `FOPEN` error in accessing a file. Prior to Release 4.5, valid values for this item were in the range 0..255. Now because there is an increase in the number of errors the file system must report, the last FOPEN error is kept in the form of an HPE status in a process structure. This HPE status gets converted to an MPE error when the user calls FCHECK. Therefore, when getting this item, the HPE status will be converted to an MPE error, and on a PUT, the MPE error will be converted to a HPE status. This is to maintain compatibility for existing applications. Currently, if you pass in an invalid FOPEN error which cannot be converted, the file system will return an FOPEN error of -20 (Invalid Operation).
2076	**Last KOPEN error (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the last `KOPEN` error in accessing a KSAM file. It is a CM `KOPEN` error status. Values are in the range 0..255.
2077	**CM arithmetic trap enabled? (B) Put: No; Verify: Yes; Release 3.0** Returns true if arithmetic traps in CM are enabled, and false otherwise.
2078	**CM arithmetic trap handler (I32) Put: No; Verify: Yes; Release 3.0** Returns a short pointer which is actually a CM Plabel. It is the plabel for the trap handler to be invoked in case of an arithmetic trap.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2079	**NM arithmetic trap mask (I32) Put: No; Verify: Yes; Release 3.0** Returns the mask for arithmetic traps raised in NM. This mask is set by the compiler. The bits and their meanings are as follows: Bits (0:7) Reserved (set to zero) Bit (8:1) Paragraph stack overflow Bit (9:1) Unimplemented error conditions Bit (10:1) Software detected misaligned result of pointer arithmetic or error in conversion from long pointer to short pointer Bit (11:1) Software detected nil pointer Bit (12:1) Range errors Bit (13:1) IEEE floating-point, invalid operation Bit (14:1) IEEE floating-point divide by zero Bit (15:1) IEEE floating-point overflow Bit (16:1) IEEE floating-point underflow Bit (17:1) IEEE floating-point inexact result Bit (18:1) Decimal divide by 0 Bit (19:1) Reserved for future use (set to 0) Bit (20:1) Unused Bit (21:1) Invalid decimal digit Bit (22:1) Invalid ASCII digit Bit (23:1) Decimal overflow Bit (24:1) 3000 mode double precision divide by zero Bit (25:1) 3000 mode double precision underflow Bit (26:1) 3000 mode double precision overflow Bit (27:1) Integer overflow Bit (28:1) 3000 mode floating-point overflow Bit (29:1) 3000 mode floating-point underflow Bit (30:1) Integer divide by zero Bit (31:1) 3000 mode floating-point divide by zero Consult the intrinsic `HPENBLTRAP` for further details.
2080	**NM arithmetic trap handler (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of the exception handler to be invoked in case of a floating-point exception.
2081	**CM library trap handler (I32) Put: No; Verify: Yes; Release 3.0** Returns a short pointer that is actually a CM plabel. It is the plabel for the trap handler to be invoked in case of a library trap in CM.
2082	**NM library trap handler (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of the trap handler to be invoked in case of a library trap in NM.
2083	**CM system trap handler (I32) Put: No; Verify: Yes; Release 3.0** Returns a short pointer which is actually a CM plabel. It is the plabel for the trap handler to be invoked in case of a system trap in CM.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2084	**NM system trap privileged level (I32) Put: No; Verify: Yes; Release 3.0** Returns the privileged level at which the trap handler executes.
2085	**NM system trap handler (@64) Put: No; Verify: Yes; Release 3.0** Returns the address of the trap handler to be invoked in case of a system trap in NM.
2086	**Unsatisfied reference handler (I32) Put: No; Verify: Yes; Release 3.0** Returns an NM plabel for the procedure to be invoked in case of an unresolved external call. By default, a load fails in cases of unresolved externals. However, if the `UNSAT` option is specified at process creation time, this item is initialized and this procedure is called instead of unresolved externals. A nil pointer indicates that there are no unresolved externals.
2087	**Unsatisfied reference procedure (CA32) Put: No; Verify: Yes; Release 3.0** Returns the name of the procedure to be invoked in case of an unsatisfied external reference at run time. The name will be left-justified and padded with blanks. It is set through the `UNSAT` option in the `RUN` command or the `CREATEPROCESS` intrinsic. It is blank if there are no unsatisfied external references in the image.
2088	**Dump armed? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies whether the `SETDUMP` intrinsic has been called by the process. True if `SETDUMP` has been called and false otherwise. If it is true, then upon process abort, Debug is called with a command string that results in a full stack trace of both the CM and the NM data stacks, and a dump of NM registers. This output is sent to the standard list device.
2089	**DEBUG commands (CA256) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the commands to be executed by Debug upon invocation when a process aborts and if a call to `SETDUMP` intrinsic had been made prior to the abort. The names are returned left-justified and padded with blanks. By default, a call to `SETDUMP` causes the Debug commands to be `tr d,i;c`. Blanks are returned if `SETDUMP` was not invoked. This item should consist only of valid Debug commands separated by a semicolon (;), the same as the macros for DAT/Debug/SAT and the `SETDUMP` intrinsic.
2090	**DEBUG armed? (B) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies whether or not Debug should be invoked if the program aborts. By default it is false. It is true if the `RUN` command is invoked with the `DEBUG` option.
2091	**CPU time, in ticks (REC) Put: No; Verify: Yes; Release 3.0** Returns the CPU time, in ticks, used by the process. It is processor dependent and accessed speedily. For processor-independent time, use item 2092 to obtain the time in milliseconds. Record type: `longint_type` (Refer to appendix B.)

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2092	**CPU time, in milliseconds (REC) Put: No; Verify: Yes; Release 3.0** Returns the CPU time, in milliseconds, used by the process. Record type: `longint_type` (Refer to appendix B.)
2093	**Does process have a SIR? (B) Put: No; Verify: Yes; Release 3.0** Returns true if this process is currently holding a SIR, and false otherwise.
2094	**JS key (I32) Put: No; Verify: Yes; Release 4.0** An internal key used to access job/session information through `AIFJSGET`. A JS key should be used only as an input key in calls to `AIFJSGET`, and should not be interpreted as any sort of job/session identifier. (JS keys are also returned by `AIFSYSWIDEGET`.)
2095	**User and file access capabilities (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the capability mask for this process. It is a bit map, and if the bit is set to 1, the process owns the corresponding capability. Bits and their meanings are as follows: Bits (0:16) Unused Bit (16:1) System manager Bit (17:1) Account manager Bit (18:1) Account librarian Bit (19:1) Group librarian Bit (20:1) Diagnostician Bit (21:1) System supervisor Bit (22:1) Create volume sets Bit (23:1) Use private volumes Bit (24:1) Use user logging Bit (25:1) Unused Bit (26:1) Programmatic sessions Bit (27:1) Network administrator Bit (28:1) Node manager Bit (29:1) Use comm subsystem Bit (30:1) Nonshareable device Bit (31:1) Save files
2096	**Time process on ready queue after awakening (REC) Put: No; Verify: Yes; Release 4.0** Returns time when process is inserted in Ready Queue. Used by Measurement Interface to calculate time spent on Ready Queue by subtracting Ready Queue time from current time (time when process is launched). This value is updated when the Measurement Interface is turned on. Record type: "longint_type" (Refer to appendix B.)
2105	**NM stack maximum SP (@64) Put: No; Verify: Yes; Release 4.0** Returns a pointer to the NM stack maximum. When a stack is initialized for a process this is the same value as the NM stack limit. It takes into account the size specified at process creation time or in the NMSTACK option in the RUN command.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2106	**Execution Mode (B) Put: No; Verify: Yes; Release 4.0** Returns the execution mode of the specified process. Note that the process' execution changes dynamically, therefore this is just a "snapshot" of the process' execution mode. True indicates the execution mode is CM, a false value indicates NM execution.
2107	**CM Maxdata (I32) Put: No; Verify: Yes; Release 4.0** Returns the maximum CM stack area size in 16 bit words.
2108	**CM S (I32) Put: No; Verify: Yes; Release 4.0** Returns the current CM top of stack in DB relative CM (16-bit) words.
2109	**JDT DST (I32) Put: No; Verify: Yes; Release 4.0** Returns the DST number of the segment for the Job Directory Table(JDT).
2110	**Job name (CA16) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the identifier given to a job or session. It must be left-justified, all capitals, and padded with blanks. All blanks represent a job or session that does not have a job name. Only the first eight characters are changed using `AIFJSPUT`. This information is local to the process.
2111	**User name (CA16) Put: No; Verify: Yes; Release 4.5** Returns the name of the user that the job or session is logged on to. It is left-justified and padded with blanks. This information is local to the process.
2112	**Group name (CA16) Put: No; Verify: Yes; Release 4.5** Returns the name of the group that the job or session is logged on to. This is left-justified and padded with blanks. This information is local to the process.
2113	**Account name (CA16) Put: No; Verify: Yes; Release 4.5** Returns the name of the account that the job or session is logged on to. This is left-justified and padded with blanks. This information is local to the process.
2114	**Maximum Account Job Priority (I32) Put: No; Verify: Yes; Release 4.5** Returns or modifies a priority that is the maximum allowed for the account that the job or session is logged on to. This information is local to the process. The maximum priority for an account is specified by using the `MAXPRI` parameter of the `NEWACCT` and `ALTACCT` commands. This item will not return valid data for jobs in the `WAIT` state. The values and their associated queues are as follows: 100 BS queue 150 CS queue 200 DS queue 250 ES queue

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2115	**Account Security (I32) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the security mask for the account that the job or session is logged on to. This information is local to the process. The account security mask can also be set using the `ALTSEC` command. Not valid for jobs in the `WAIT` state. The bits of the mask have the following meanings: Bits (0:20) Unused Bit (20:1) Read any Bit (21:1) Read account user Bit (22:1) Append any Bit (23:1) Append account user Bit (24:1) Write any Bit (25:1) Write account user Bit (26:1) Lock any Bit (27:1) Lock account user Bit (28:1) Execute any Bit (29:1) Execute account user Bits (30:2) Unused

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2116	**Group security (I32) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the security mask for the group that the job or session is logged on to. This information is local to the process. The group security mask can also be set using the `ALTSEC` command. Not valid for jobs in the `WAIT` state. The bits of the mask have the following meanings: Bits (0:2) Unused Bit (2:1) Read any Bit (3:1) Read account user Bit (4:1) Read account librarian Bit (5:1) Read group user Bit (6:1) Read group librarian Bit (7:1) Append any Bit (8:1) Append account user Bit (9:1) Append account librarian Bit (10:1) Append group user Bit (11:1) Append group librarian Bit (12:1) Write any Bit (13:1) Write account user Bit (14:1) Write account librarian Bit (15:1) Write group user Bit (16:1) Write group librarian Bit (17:1) Lock any Bit (18:1) Lock account user Bit (19:1) Lock account librarian Bit (20:1) Lock group user Bit (21:1) Lock group librarian Bit (22:1) Execute any Bit (23:1) Execute account user Bit (24:1) Execute account librarian Bit (25:1) Execute group user Bit (26:1) Execute group librarian Bit (27:1) Save any Bit (28:1) Save account user Bit (29:1) Save account librarian Bit (30:1) Save group user Bit (31:1) Save group librarian
2117	**Home Group (CA16) Put: No; Verify: Yes; Release 4.5** Returns the name of the home group for the user that the job or session is logged on to. This is left-justified and padded with blanks. Not valid for jobs in the `WAIT` state. This information is local to the process.
2118	**Account Local Attributes (I32) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the local attributes for the account that the job or session is currently logged on to. These attributes are an extension of MPE/iX security and are not required. Their meaning is user defined, although the first 16 bits are unused. Not valid for jobs in the `WAIT` state. This information is local to the process.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2119	**User Capabilities (I32) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the user capability mask for the job or session. Not valid for jobs in the `WAIT` state. This information is local to the process. Mask bits and their meanings are as follows: Bits (0:16) Unused Bit (16:1) System manager Bit (17:1) Account manager Bit (18:1) Account librarian Bit (19:1) Group librarian Bit (20:1) Diagnostician Bit (21:1) System supervisor Bit (22:1) Create volume sets Bit (23:1) Use private volumes Bit (24:1) Use user logging Bit (25:1) Unused Bit (26:1) Programmatic session Bit (27:1) Network administrator Bit (28:1) Node manager Bit (29:1) Use comm subsystem Bit (30:1) Non-shareable device Bit (31:1) Save files
2120	**General Resource Capabilities (I32) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the general resources capability mask for the job or session. This mask contains the general resource capabilities for the user that the job or session is logged on to. Not valid for jobs in the `WAIT` state. This information is local to the process. Mask bits and their meanings are as follows: Bits (0:23) Unused Bit (23:1) Batch access Bit (24:1) Interactive access Bit (25:1) Privileged mode Bit (26:2) Unused Bit (28:1) Multiple RINs Bit (29:1) Unused Bit (30:1) Extra data segment Bit (31:1) Process handling

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description

2121	**Allow Mask (BA96) Put: Yes; Verify: Yes; Release 4.5**
	
	Returns or modifies the commands allowed for this session in a packed array of 96 booleans. True is returned if the command is allowed for the job or session. Not valid for jobs in the `WAIT` state. This information is local to the process. The commands and their array locations are as follows:
	
	`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  STOPSPOOL = 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`

2122	**Pathnames of Open Files (@64) Put: No; Verify: No; Release 4.5** Returns a list of file pathnames for each file opened by this process. The long pointer specified in this item points to the buffer where the names will be returned. Each name will be terminated by a NULL character. On input, the first four bytes in the buffer will represent the maximum buffer length in bytes. On output, the first four bytes will contain the actual number of bytes returned (this includes the NULL terminators separating each name). Refer to Operation Notes for `AIFSYSWIDEGET` for a diagram of the buffer format. If an HFS file has been opened by UFID and not by name or if the file is a special MPE file which cannot be represented as an HFS pathname (for example, $TEMPDIRC), then there will be no pathname associated with the file. In this case, a NULL character will be written to the buffer to delimit a place holder for the file. This is so that each name will match up with a corresponding entry in the file number and path_identifier arrays (items 2063 and 2123). See the buffer_type declaration in appendix B for a suggestion on how to define this buffer.
2123	**Path Identifiers of Open Files (RECA) Put: No; Verify: Yes; Release 4.5** Returns the unique path identifiers for the open files. If a file has been opened by UFID and not by name or if the file is a special MPE file not supported by the hierarchical directory services, then the path_identifier entry for that file will not contain all the information needed to identify a unique pathname. The parent_ufid and link ID fields will be 0. Record type: path_id_rec_type

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2125	**Fork Process (B) Put: No; Verify: Yes; Release 4.5** Returns true if the process was created using the `fork()` function.
2126	**UID (I32) Put: Yes; Verify: Yes; Release 4.5** Returns the process' real user ID. This POSIX attribute is assigned to each user on the system. Every process created by a user has that user's real user ID.
2127	**EUID - Effective UID (I32) Put: Yes; Verify: Yes; Release 4.5** Returns the process' effective user ID. This POSIX attribute describes the access rights of a process to files. At process creation time, the EUID matches the UID, but it may be changed with the `setuid()` function.
2128	**GID (I32) Put: Yes; Verify: Yes; Release 4.5** Returns the process' real group ID. This POSIX attribute describes the group a process belongs to.
2129	**EGID - Effective GID (I32) Put: Yes; Verify: Yes; Release 4.5** Returns the process' effective Group ID. This is a POSIX attribute which determines access rights of a process to files. At process creation time, the EGID matches the GID, but it may be changed with the `setgid()` function.
2130	**CMASK (U32) Put: Yes; Verify: Yes; Release 4.5** Returns the process' file creation mask. This mask changes with the POSIX `umask()` function. Any file created after a process calls umask will have an ACD attached to the file.
2131	**Program pathname (REC) Put: No; Verify: Yes; Release 4.5** Returns the absolute pathname of the program file. On input, the first four bytes in the buffer will represent the maximum buffer length in bytes. On output, the name will be terminated by a NULL character and the first four bytes will contain the actual number of bytes returned (not including the NULL character or the one word length). Note that this item should only be used for names that can be expressed using HFS syntax. If you specify this item for a program (for example, $OLDPASS) that cannot be expressed using HFS syntax, then a warning is returned in itemstatus_array. Record type: pathname_type
2132	**Break Request Done (B) Put: No; Verify: Yes; Release 4.5** Returns the Boolean flag used to indicate whether a process received the process management break interrupt message or not. Set to true to indicate that the process has been put into a break wait. Set to false to indicate that the process is no longer in break wait but on the dispatcher queue.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2133	**Break Request Cancel (I32) Put: No; Verify: Yes; Release 4.5** Returns counter used by process management to indicate whether a pending process management break interrupt message just received is to be ignored or not.
2134	**Break Request Pending (I32) Put: No; Verify: Yes; Release 4.5** Returns counter used by process management to indicate whether a process management break interrupt message that is being sent has been received and is being acted upon. The process receiving the message will be in a break wait state.
2135	**List of sibling PIDs (REC) Put: No; Verify: No; Release 4.5** Returns a list of PIDs of all the sibling processes. The maximum size is a system constant, obtainable from AIFSCGET. You should pass a buffer of appropriate size. The first word of the buffer is expected to hold the size, in longwords, of the rest of the buffer area. The first word, upon return, specifies the number of PIDs returned. Check itemstatus_array to see if information was truncated. Record type: I64rec_type (Refer to appendix B)
2136	**List of parent PIDs (REC) Put: No; Verify: No; Release 4.5** Returns a list of PIDs of all the parent processes. The maximum size is a system constant, obtainable from AIFSCGET. You should pass a buffer of appropriate size. The first word of the buffer is expected to hold the size, in longwords, of the rest of the buffer area. The first word, upon return, specifies the number of PIDs returned. Check itemstatus_array to see if information was truncated. Record type: I64rec_type (Refer to appendix B)
2142	**Interactive? (B) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the interactive status of the process. True when human intervention is required for all input operations.
2143	**Environment Nil (B) Put: No; Verify: Yes; Release 5.0** Returns whether the user has a nil environment. If a process has a nil environment, then during startup of its child POSIX process, all the CI variables for the job/session environment are changed into an environment format and inherited by the child process.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2144	**Workgroup name (CA256) Put: Yes; Verify: Yes** Returns or modifies the name of the workgroup to which the specified process belongs. The workgroup name will be left justified and terminated by a NULL character (ASCII 0). AIFPROCPUT with this item number will move the target process to the specified workgroup. A process moved in this manner is considered an **artificial** member of the workgroup (the process was placed in a workgroup explicitly, rather than naturally by meeting the membership criteria specified for the workgroup). A process remains an artificial member of its assigned workgroup until either the workgroup is purged or its explicit assignment is changed by AIFPROCPUT item 2146. An artificial member is not affected by a system-wide scan or by the changing of its process attributes used to determine workgroup membership. If both items 2144 and 2146 are specified, then the order they are passed will determine the outcome. If the order is item 2144 followed by item 2146, then the process will migrate to its natural workgroup. However if item 2146 is passed followed by item 2144, then the process will be pegged to the name of the workgroup passed as its artificial member. Record type: `CA256`
2145	**Artificial Member? (B) Put: No; Verify: Yes** Returns true if the specified process is an artificial member of its workgroup.
2146	**Return to Natural Workgroup (B) Put: Yes; Verify: No** Returns the specified process to its natural workgroup. This item will return a warning for AIFPROCGET in the item status array. This item allows the process to migrate to its natural workgroup. If the process is an artificial member of a workgroup, this item will release the process from its explicit workgroup assignment. However a process who is a member of its natural workgroup will not be impacted. If both items 2144 and 2146 are specified, then the order they are passed will determine the outcome. If the order is item 2144 followed by item 2146, then the process will migrate to its natural workgroup. However if item 2146 is passed followed by item 2144, then the process will be pegged to the name of the workgroup passed as its artificial member.
2147	**Execution State? (I32) Put: No; Verify: Yes** Returns the execution state of the process. Values and their meanings are: 0 Blocked for memory manager. 1 Unused. 2 Blocked for control block. 3 All purpose block, usually waiting for a message. 4 Ready to execute (or executing). 5 Blocked for terminal write or control.

Table 3-27. Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
2148	**Fixed Priority? (I32) Put: Yes; Verify: NO** Fixes the priority of the process at the value passed. This item will return a warning for AIFPROCGET. A valid MPE/iX priority is in the range 0 .. 32767. This priority can be mapped to MPE V priority by the following formula: MPEVPri= (32767 - MPEXLPri) DIV 128 (All formula values are decimal.)

AIFREPLYGET

Returns information on a specified pending reply request.

Syntax

```
                    REC             I32A            @64A
AIFREPLYGET (overall_status, itemnum_array, item_array,
                    RECA            I32             I32
             itemstatus_array, reply_request_id, user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

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

Passes the request ID that uniquely identifies the reply request to be retrieved. If the value passed in ***reply_request_id*** is greater than the total number of allocated requests on the system, an error is returned in ***overall_status***.

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFREPLYGET` requires only a reply request ID as the input parameter. A list of reply request IDs may be obtained by calling `AIFSYSWIDEGET` with area 14000.

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with reply information returned by `AIFREPLYGET`.

Table 3-28. Reply Information Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
14001	**Is entry active? (B) Release 3.0** Returns true if the reply request is active, and false when the reply request is inactive.
14002	**Process type (I32) Release 3.0** Returns the type of the process that requested the reply. Values and their meanings are as follows: 1 System process 2 User process
14003	**Creation time (I32) Release 3.0** Returns the time (hours/minutes/seconds/tenths of seconds) when the reply request was created. All fields are 0's if the request is inactive. The format returned in the 32-bit integer is the same as that returned by the `CLOCK` intrinsic. The bits and their meanings are as follows: Bits (0:8) The hour of the day Bits (8:8) The minute of the hour Bits (16:8) The seconds Bits (24:8) The tenths of seconds
14004	**Job/session number (I32) Release 3.0** Returns the job/session number for the job or session that requested the reply. Valid only for user processes. A zero is returned for system processes. The format of the job/session number is as follows: Bits (0:2) Job or session? (1 = Session, 2 = Job) Bits (2:30) Job or session number
14005	**Reply request ID (I32) Release 3.0** Returns the reply request ID of the store/restore activity. Valid only for user processes. A zero is returned for system processes.
14006	**Message text (CA160) Release 3.0** Returns the text portion of the reply request, as it is normally displayed on the console. The set and message numbers of the reply request are used to fetch the message from the message catalog. Parameters (refer to item 14011) may be inserted into the message wherever a "!" is found.
14007	**Message source (I32) Release 3.0** Returns a value indicating the source of the message text. Values and their meanings are as follows: 1 Message catalog 2 Supplied literal

Table 3-28. Reply Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
14008	**Message length (I32) Release 3.0** Returns the length of the message returned in item 14006.
14009	**Request set number (I16) Release 3.0** Returns the message set number in the message catalog for the specified request. Values and their meanings are as follows: >0 Message number within message set -1 String is passed in, rather than found in, the message catalog
14010	**Request message number (I16) Release 3.0** The return depends on whether the message set number is greater than zero or less than zero: ■ If the message set number is greater than zero (see item 14009), returns the message number in the message catalog for the specified request . ■ If the message set number is less than zero, returns the byte address of the string that is passed in.
14011	**Parameter (CA80) Release 3.0** Returns the parameters whose types are defined by item 14012 "Parameter type." A message can contain up to five parameters. The parameters are inserted wherever an "!" is found in a message. Parm1 substitutes for the leftmost parameter in the message, parm2 for the next parameter to the right, and so on. If parm(*n*) is present, parm(*n*-1) must also be present. If parm is a string, a byte address must be passed.
14012	**Parameter type (REC) Release 3.0** Returns values indicating the data types of any parameters that are returned in item 14011. Values indicating data types are: 0 Parm is the address of a byte array, terminated by a null (0) 1 Parm is a 16-bit integer value 2 Parm is the address of a 32-bit integer 3 Parm should be ignored The bits where each of the parameter type indicators are located are: Bits (0:1) If set to 1, ignore all parameters Bits (1:3) Type of parm1 Bits (4:3) Type of parm2 Bits (7:3) Type of parm3 Bits (10:3) Type of parm4 Bits (13:3) Type of parm5 Record type: `bit16` (Refer to appendix B.)

AIFSCGET

Returns system configuration information

Syntax

```
                                REC          I32A
AIFSCGET(overall_status,  itemnum_array,
                              @64A           RECA
                         item_array,  itemstatus_array,
                           I32
                         user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the information to be returned to a data structure pointed to in the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

`AIFSCGET` does not require any specific inputs because the information returned is global to the system.

AIFSCPUT

Modifies system configuration information

Syntax

```
                                REC          I32A
AIFSCPUT(overall_status,  itemnum_array,
                               @64A           RECA
                          item_array,  itemstatus_array,
                             I32          I32A
                          user_id,  ver_item_nums,
                             @64A           RECA
                          ver_items,  ver_item_statuses )
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure containg new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in ver_items. If n items are being verified, element n+1 must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in ver_item_nums.

Array type: globalanyptr

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in ver_items. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

Default: nil

Operation Notes

SYSGEN will not reflect any changes made dynamically by AIFSCPUT to the system logging mask because it gets its information from the system configuration files and not the system tables that have been changed.

AIFSCGET/PUT Items

The following two tables provide summary and detailed descriptions of the items numbers associated with system configuration information.

Item Summary

The following table summarizes the item numbers associated with system configuration information. For more detailed information about these item numbers, refer to the table of system configuration information item descriptions.

Table 3-29. System Configuration Information 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 number	Y	Y	1	16383	-3008
3007	I32	Next session number	Y	Y	1	16383	-3009
3008	I32	Job security	Y	Y	0	3	-3010
3009	B	Single user?	N	Y			
3010	B	Out of resources?	N	Y			
3011	B	Out of LDEVs?	N	Y			
3012	B	Low on disk?	N	Y			
3013	I32	Logical console	N	Y			
3014	I32	Physical console	N	Y			
3015	BA96	Global 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
3028	I32	ES queue limit	Y	Y	127	13567	-3012
3029	I32	CS quantum maximum	Y	Y			
3030	I32	CS quantum minimum	Y	Y			
3031	I32	DS quantum	Y	Y			
3032	I32	ES quantum	Y	Y			
3033	I32	CS quantum	N	Y			

Table 3-29. System Configuration Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
3034	I32	Maximum open files	N	Y			
3035	I32	Maximum processes	N	Y			
3036	I32	Maximum job/sessions	N	Y			
3037	CA8	MPE/iX version ID	N	Y			
3038	I32	Serial number	N	Y			
3039	I32	Memory size	N	Y			
3040	I32	Total DST entries	N	Y			
3041	I32	Available DST entries	N	Y			
3042	I32	Rounding factor	N	Y			
3043	I32	Tick/msec conversion factor	N	Y			
3044	CA8	AIF:MI version ID	N	Y			
3045	CA8	AIF:OS version ID	N	Y			
3046	I32	Cold load ID	N	Y			
3047	I32	Current PIN highwater mark	N	Y			
3048	I32	Maximum LDEV number	N	Y			
3049	I32	CS Boost property	Y	Y	0	1	-3015
3050	I32	CS Queue timeslice	Y	Y			
3051	I32	DS Boost property	Y	Y	0	1	-3015
3052	I32	DS Queue timeslice	Y	Y			
3053	I32	ES Boost property	Y	Y	0	1	-3015
3054	I32	ES Queue timeslice	Y	Y			
3055	I32	Maximum Job Limit	N	Y			
3056	I32	Maximum Session Limit	N	Y			
3057	CA8	MPE Release VUF	N	Y			
3058	CA8	MPE User VUF	Y	Y			
3059	I32	Max Number of Processors	N	Y			
3060	B	Autoboot Toggle	Y	Y			
3061	I32	Actual Num of Processors	N	Y			
3062	CA256	Logon Prompt	Y	Y			
3063	I32	Default NM Stack	N	Y			
3064	I32	Maximum NM Stack	N	Y			
3065	I32	Default CM Stack	N	Y			
3066	I32	Maximum CM Stack	N	Y			
3067	I32	Default Heap	N	Y			
3068	I32	Maximum NM Heap	N	Y			
3069	I32	Maximum number of AIF ports	N	Y			
3070	I32	Maximum Path Length	N	Y			

Table 3-29. System Configuration Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
3071	CA80	Machine type	N	Y			
3072	CA256	Network node name	N	Y			
3073	B	Password encryption	N	Y			
3074	I32	Minimum password length	N	Y			
3075	I32	Maximum invalid logons per device	N	Y			
3076	B	Password prompt required	N	Y			
3077	B	UDC failure termination	N	Y			
3078	B	Minimum assistance logon	N	Y			
3079	B	Fopen logging extension	N	Y			
3080	I32	Idle session termination	N	Y			
3081	I32	Down device timeout	N	Y			
3082	B	Programmatic command disabling warning	N	Y			
3083	I32	Password expiration interval in days	N	Y			
3084	U32	Next global password expiration date	N	Y			
3085	I32	Password expiration warning	N	Y			
3086	I32	Embedded password disallow	N	Y			
3087	I32	Cross stream restriction and authorization	N	Y			
3088	I32	Stream privilege and authorization	N	Y			
3089	B	Assurance of auditability	N	Y			
3090	B	Maximum file protection	N	Y			
3091	I32	Global user password maximum days	N	Y			
3092	I32	Global user password minimum days	N	Y			
3093	I32	Global user password expiration days	N	Y			
3094	I32	Global user password warning days	N	Y			
3095	I32	Maximum invalid user logons	N	Y			
3096	I32	Disabled user timeout	N	Y			
3097	B	Security installed	N	Y			

Table 3-29. System Configuration Information Item Summary (continued)

Item	Type	Description	Put	Ver	Min	Max	Error#
3099	I32	Number of currently configured workgroups	N	Y			
3100	B	Purge Scan	Y	Y			
3101	B	System-wide scan pending?	N	Y			
3112	I32	Workgroup creation count	N	Y			
3102	I32	Lower job number limit	Y	Y	1	16383	-3023
3103	I32	Upper job number limit	Y	Y	0	16383	-3024
3104	I32	Lower session number limit	Y	Y	1	16383	-3023
3105	I32	Upper session number limit	Y	Y	0	16383	-3024
3106	I32	Lower input spoolid limit	Y	Y	1	9999999	-3023
3107	I32	Next input spoolid	Y	Y	1	9999999	-3025
3108	I32	Upper input spoolid limit	Y	Y	0	9999999	-3024
3109	I32	Lower output spoolid limit	Y	Y	1	9999999	-3023
3110	I32	Next output spoolid	Y	Y	1	9999999	-3026
3111	I32	Upper output spoolid limit	Y	Y	0	9999999	-3024

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with system configuration information.

Table 3-30. System Configuration Information Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Description
3001	**Job fence (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current jobfence on the system, a value in the range 0..14. If a job's input priority (INPRI) is higher than the system jobfence, that job attempts to log on. The jobfence can also be set using the `JOBFENCE` command.
3002	**Job limit (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of jobs allowed to be concurrently logged on to the system, a value from 0 to the value configured in SYSGEN. The job limit can also be set using the `LIMIT` command.
3003	**Job count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of jobs that currently exist on the system.
3004	**Session limit (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of sessions allowed to be concurrently logged on to the system, a value from 0 to the value configured in SYSGEN. The session limit can also be set using the `LIMIT` command.
3005	**Session count (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of sessions that currently exist on the system.
3006	**Next job number (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the number assigned to the next job that is streamed, a value in the range 1..16383. Do not set this value to that of an existing job number. You may set a value outside the range defined by the lower and upper job number limits, but it will not be used unless the limits are changed such that the value is in range before the system needs to assign it to a job. This is because the system always tries to assign a value within limits. See the `SETCOUNTER` command description for further details.
3007	**Next session number (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the number assigned to the next session that successfully logs on, a value in the range 1..16383. Do not set this value to that of an existing session number. You may set a value outside the range defined by the lower and upper session number limits, but it will not be used unless the limits are changed such that the value is in range before the system needs to assign it to a session. This is because the system always tries to assign a value within limits See the `SETCOUNTER` command description for further details.
3008	**Job security (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current job security with the following values: 0 Job security high 3 Job security low 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. Job security can also be set using the `JOBSECURITY` command.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3009	**Single user mode (B) Put: No; Verify: Yes; Release 3.0** Returns true if the system is in single-user mode, and false if the system is in multiuser mode.
3010	**Out of resources (B) Put: No; Verify: Yes; Release 3.0** Returns true if 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.
3011	**Out of LDEVs (B) Put: No; Verify: Yes; Release 3.0** Returns true if a job has failed to log on because the specified output device is unavailable. This flag is set to false when that device becomes available.
3012	**Low on disk space (B) Put: No; Verify: Yes; Release 3.0** Returns true if there is not enough disk space for a logon to succeed. This flag is set to false if there is enough disk space for a successful logon.
3013	**Logical console LDEV (I32) Put: No; Verify: Yes; Release 3.0** Returns the current logical console's LDEV number. The logical console LDEV number can be changed using the `CONSOLE` command.
3014	**Physical console LDEV (I32) Put: No; Verify: Yes; Release 3.0** Returns the physical console's LDEV number.

3015	**Global allow mask (BA96) Put: Yes; Verify: Yes; Release 3.0**
	Returns or modifies a packed array of 96 booleans indicating the commands allowed for this system. True indicates that the corresponding command is allowed. Following are the commands and their array locations:
	`ABORTIO = 1` `DELETESPOOLFILE = 19` `LDISMOUNT = 37`
	`ACCEPT = 2` `DISALLOW = 20` `MRJECONTROL = 38`
	`DOWN = 3` `JOBFENCE = 21` `JOBSECURITY = 39`
	`GIVE = 4` `LIMIT = 22` `DOWNLOAD = 40`
	`HEADOFF = 5` `STOPSPOOL = 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 CONTROL = 45`
	`TAKE = 10` `RESUMESPOOL = 28` `SHOWCOM = 46`
	`UP = 11` `STREAMS = 29` `OPENQ = 47`
	`MPLINE = 12` `CONSOLE = 30` `SHUTQ = 48`
	`DSCONTROL = 13` `WARN = 31` `DISCRPS = 49`
	`ABORTJOB = 14` `WELCOME = 32` `VSRESERVESYS = 50`
	`ALLOW = 15` `MON = 33` `VSRELEASESYS = 51`
	`ALTSPOOLFILE = 16` `MOFF = 34` `VSCLOSE = 52`
	`ALTJOB = 17` `VMOUNT = 35` `VSOPEN = 53`
	`BREAKJOB = 18` `LMOUNT = 36` `SPOOLER = 54`
	`unused = 55..96`

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description

3016	**System logging mask (BA64) Put: Yes; Verify: Yes; Release 3.0**
	
	Returns or modifies a packed array of 64 booleans indicating the system logging mask. True indicates that the logging type is on. Following are the logging types and their array locations:
	
	`Log failure = 0` `AIF = 30`
	`System up = 1` `Processor launch event = 31`
	`Job initiation = 2` `Password changes = 34`
	`Job termination = 3` `System logging configuration = 35`
	`Process termination = 4` `Restore logging = 36`
	`File close = 5` `Printer access failure = 37`
	`Shutdown = 6` `ACD changes = 38`
	`Power failure = 7` `Stream initiation = 39`
	`Spooling log = 8` `User logging = 40`
	`Line disconnect = 9` `Process creation = 41`
	`Line close = 10` `Security config. change = 42`
	`I/O error = 11` `Chgroup = 43`
	`Physical (dis)mount = 12` `File open = 44`
	`Logical (dis)mount = 13` `Command logging = 45`
	`Tape labels = 14` `Maintenance request log = 46`
	`Console log = 15` `Diagnostic control unit = 47`
	`Program file event = 16` `Diagnostic information = 50`
	`Call progress = 17` `High-priority machine check = 51`
	`DCE information = 18` `Low-priority machine check = 52`
	`Unused = 19` `Directory open/close logging = 55`
	`NCS spooling = 20` `CM file close = 60`
	`Unused = 21-29` `Chdir = 61`
	`Process adopt = 62`
	`Chown = 63`

3017	**Streams LDEV (I32) Put: No; Verify: Yes; Release 3.0** Returns the streams LDEV as currently set on the system. The streams LDEV can also be set using the `STREAMS` command.
3018	**System outfence (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the system outfence, a value in the range 1..14. A value of 14 prevents all spool files on the system from being printed. The system outfence can also be set using the `OUTFENCE` command.
3019	**AS queue base (I32) Put: No; Verify: Yes; Release 3.0** Returns 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 by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3020	**AS queue limit (I32) Put: No; Verify: Yes; Release 3.0** Returns 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 by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3021	**BS queue base (I32) Put: No; Verify: Yes; Release 3.0** Returns 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 by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3022	**BS queue limit (I32) Put: No; Verify: Yes; Release 3.0** Returns 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 by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3023	**CS queue base (I32) Put: No; Verify: Yes; Release 3.0** Returns 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 by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3024	**CS queue limit (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the CS queue limit. This value is the lowest priority that any process in the CS queue can have. It can also be set using the `TUNE` command. This priority can be mapped to MPE V by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3025	**DS queue base (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the base of the DS queue. This value is the highest priority that any process in the DS queue can have. It can also be set using the `TUNE` command. This priority can be mapped to MPE V by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3026	**DS queue limit (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the DS queue limit. This value is the lowest priority that any process in the DS queue can have. It can also be set using the `TUNE` command. This priority can be mapped to MPE V by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3027	**ES queue base (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the base of the ES queue. This value is the highest priority that any process in the ES queue can have. It can also be set using the `TUNE` command. This priority can be mapped to MPE V by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3028	**ES queue limit (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the ES queue limit. This value is the lowest priority that any process in the ES queue can have. It can also be set using the `TUNE` command. This priority can be mapped to MPE V by the following formula: `MPEVPri = (32767 - MPEiXPri) DIV 128  (All formula values are decimal.)`
3029	**CS quantum maximum (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of milliseconds that a process in the CS queue may execute before its priority starts decaying.
3030	**CS quantum minimum (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the minimum number of milliseconds that a process in the CS queue must execute before its priority may be reduced.
3031	**DS quantum (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of milliseconds that a process in the DS queue may run before its priority starts decaying. This can also be set using the `TUNE` command.
3032	**ES quantum (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of milliseconds that a process in the ES queue may run before its priority starts decaying. This can also be set using the `TUNE` command.
3033	**CS quantum (I32) Put: No; Verify: Yes; Release 3.0** Returns the average number of milliseconds that a process in the CS queue executes before it is 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) Put: No; Verify: Yes; Release 3.0** Returns the maximum number of files a process can have open at the same time.
3035	**Maximum number of processes (I32) Put: No; Verify: Yes; Release 3.0** Returns the maximum number of processes that can be executing on the system at the same time.
3036	**Maximum number of jobs/sessions (I32) Put: No; Verify: Yes; Release 3.0** Returns the maximum number of jobs and sessions that can be configured by SYSGEN. The combined number of jobs and sessions configured by SYSGEN cannot exceed this number.
3037	**MPE/iX version ID (CA8) Put: No; Verify: Yes; Release 3.0** Returns the version ID (VUF) for the MPE/iX operating system on the machine.
3038	**Serial number (I32) Put: No; Verify: Yes; Release 3.0** Returns the machine's serial number (CI variable `HPSUSAN`).
3039	**Memory size (I32) Put: No; Verify: Yes; Release 3.0** Returns the memory size, interpreted as the number of 2-K pages.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3040	**Total number of DST entries (I32) Put: No; Verify: Yes; Release 3.0** Returns the total number of DST entries on the system.
3041	**Available number of DST entries (I32) Put: No; Verify: Yes; Release 3.0** Returns the total number of unused DST entries on the system.
3042	**Rounding factor (I32) Put: No; Verify: Yes; Release 3.0** Timer counters should be multiplied by this value to convert them to ticks.
3043	**Tick to millisecond conversion factor (I32) Put: No; Verify: Yes; Release 3.0** Returns the number of ticks in one millisecond for the machine.
3044	**AIF:MI version ID (CA8) Put: No; Verify: Yes; Release 3.0** Returns the Architected Interface Facility: Measurement Interface product version that is on the machine.
3045	**AIF:OS version ID (CA8) Put: No; Verify: Yes; Release 3.0** Returns the Architected Interface Facility: Operating System product version that is on the machine.
3046	**Cold load ID (I32) Put: No; Verify: Yes; Release 3.0** Returns a value that is increased each time the machine is booted.
3047	**Current PIN Highwater Mark (I32) Put: No; Verify: Yes; Release 3.0** Returns the current highwater PIN number in use. The maximum value that can ever be returned in this field can be found by item #3035.
3048	**Maximum LDEV number (I32) Put: No; Verify: Yes; Release 3.0** Returns the maximum logical device number that can be configured on the machine.
3049	**CS boost property (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current boost property of the CS queue. Values and their meanings are as follows: 0 Decay 1 Oscillate
3050	**CS queue timeslice (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of milliseconds that a CPU-bound process running in the CS queue can monopolize the CPU. When a process in the CS queue has held the CPU for the timeslice, the dispatcher examines the running process and makes the necessary adjustments. This value is accurate to 100-millisecond granularity and has a minimum value of 100 milliseconds.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3051	**DS boost property (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current boost property of the DS queue. Values and their meanings are as follows: 0 Decay 1 Oscillate
3052	**DS queue timeslice (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of milliseconds that a CPU-bound process running in the DS queue can monopolize the CPU. When a process in the DS queue has held the CPU for the timeslice, the dispatcher examines the running process and makes the necessary adjustments. This value is accurate to 100 millisecond granularity, and has a minimum value of 100 milliseconds.
3053	**ES Boost property (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the current boost property of the ES queue. Values and their meanings are: 0 Decay 1 Oscillate
3054	**ES queue timeslice (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the maximum number of milliseconds that a CPU-bound process running in the ES queue can monopolize the CPU. When a process in the ES queue has held the CPU for the timeslice, the dispatcher examines the running process and makes the necessary adjustments. This value is accurate to 100 millisecond granularity, and has a minimum value of 100 milliseconds. 0 Decay 1 Oscillate
3055	**Maximum job limit(I32) Put: No; Verify: Yes; Release 4.0** Returns the maximum job limit which can be set by the operator with the :LIMIT command. The maximum job limit is set by using the command JOB MAXLIMIT=n in SYSGEN.
3056	**Maximum session limit (I32) Put: No; Verify: Yes; Release 4.0** Returns the maximum session limit which can be set by the operator with the :LIMIT command. The maximum session limit is set by using the command SESSION MAXLIMIT=n in SYSGEN.
3057	**MPE release version (CA8) Put: No; Verify: Yes; Release 4.0** Returns the MPE release version number in fhe format of V.UU.FF as shown in the SHOWME command. For example: RELEASE:A.41.00.
3058	**MPE user version (CA8) Put: Yes; Verify: Yes; Release 4.0** Returns or modifies the MPE users version number in the format of V.UU.FF as shown in the SHOWME command. For example: USER VERSION: A.41.00.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3059	**Maximum number of processors (I32) Put: No; Verify: Yes; Release 4.0** Returns the maximum number of processors allowed on the system. This number is system specific and defined by the OS.
3060	**Autoboot toggle (B) Put: Yes; Verify: Yes; Release 4.0** Returns or modifies the autoboot flag in the PDC stable storage. Stable storage has a write life of 1000 writes. After that the values can no longer be guaranteed. Use with caution.
3061	**Actual number of processors (I32) Put: No; Verify: Yes; Release 4.0** Returns the actual number of processors on the system. This is configurable at system boot up time.
3062	**Logon Prompt (CA256) Put: Yes; Verify: Yes; Release 4.5** Returns or modifies the system logon prompt.
3063	**Default NM Stack (I32) Put: No; Verify: Yes; Release 4.5** Returns the value of default NM stack configured by SYSGEN.
3064	**Maximum NM Stack (I32) Put: No; Verify: Yes; Release 4.5** Returns the value of maximum NM stack configured by SYSGEN.
3065	**Default CM Stack (I32) Put: No; Verify: Yes; Release 4.5** Returns the value of default CM stack configured by SYSGEN.
3066	**Maximum CM Stack (I32) Put: No; Verify: Yes; Release 4.5** Returns the value of maximum CM stack configured by SYSGEN.
3067	**Default Heap (I32) Put: No; Verify: Yes; Release 4.5** Returns the value of the default heap configured by SYSGEN.
3068	**Maximum NM Heap (I32) Put: No; Verify: Yes; Release 4.5** Returns the value of the maximum NM heap configured by SYSGEN.
3069	**Maximum number of AIF ports (I32) Put: No; Verify: Yes; Release 4.5** Returns the maximum number of AIF ports.
3070	**Maximum Path Length (I32) Put: No; Verify: Yes; Release 4.5** Returns the maximum path length currently allowed on the system. The length includes one byte for the Null terminator required at the end of the pathname.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3071	**Machine type (CA80) Put: No; Verify: Yes; Release 5.0** Returns the hardware type on which the system is running. For example, Series 955.
3072	**Network Node Name (CA256) Put: No; Verify: Yes; Release 5.0** Returns the node name of the system in the communication network. This is the local domain name used during NS Configuration in NMMGR.
3073	**Password Encryption (B) Put: No; Verify: Yes; Release 5.0** Returns true when global password encryption is on and false when off. When set to true, new or altered account, group, and user passwords are encrypted. The encryption feature is enabled in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3074	**Minimum Password Length (I32) Put: No; Verify: Yes; Release 5.0** Returns the minimum password length. A valid range of values is 0 to 8. The minimum password length feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3075	**Maximum Invalid Logons per Device (I32) Put: No; Verify: Yes; Release 5.0** Returns the global maximum invalid logon count per NAILED terminal. The valid range of values is 0 to 32766. The maximum invalid logons per device feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3076	**Password Prompt Required (B) Put: No; Verify: Yes; Release 5.0** Returns the global feature indicating whether all interactive attempts to initiate jobs or sessions must NOT have embedded passwords in the logon string. A value of true indicates embedded passwords are not allowed on the HELLO, STREAM, or STARTSESSION commands issued from a terminal. If passwords are present, the logon will be rejected regardless of their validity. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3077	**UDC Failure Termination (B) Put: No; Verify: Yes; Release 5.0** Returns true when session termination is enabled for an error in UDC initiation. When this option is enabled, a UDC initiation failure at the account or system level UDC will cause the job/session to be terminated except in the case where MANAGER.SYS or OPERATOR.SYS is logged on at the system console. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3078	**Minimum Assistance Logon (B) Put: No; Verify: Yes; Release 5.0** Returns true when terse error messages are enabled for errors encountered in parsing and verifying the logon command. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3079	**FOPEN Logging Extension (B) Put: No; Verify: Yes; Release 5.0** Returns true when all FOPEN calls are logged and false for FOPEN FAILURE ONLY logging. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3080	**Idle Session Termination (I32) Put: No; Verify: Yes; Release 5.0** Returns the configured length of time that a CI process will wait on a read. If there is no response during that duration, then the idle session is terminated. The valid range of values is from 0 to 546 minutes. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3081	**Down Device Timeout (I32) Put: No; Verify: Yes; Release 5.0** Returns the DOWN device timeout value in seconds. When the timeout expires for the DOWN device the Security Process will UP the device. Changing this value in the HP Security Monitor does NOT impact any timeouts currently in effect. The valid range of values is 0 to 32766 seconds. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3082	**Programmatic Command Disabling Warning (B) Put: No; Verify: Yes; Release 5.0** Returns true when programmatic access to a command causes a warning to be logged via the command logging facility. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3083	**Password Expiration Interval in Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the global password expiration interval in days. When a password expiration date arrives this interval is added to the expiration date to automatically set the next global user password expiration date. A valid range of values is 0 to 365 days. This feature is in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3084	**Next Global Password Expiration Date (U32) Put: No; Verify: Yes; Release 5.0** Returns the next global password expiration date. When this feature is enabled it expires all the REQUIRED user passwords on the same global expiration date. The format returned in the 32-bit integer is similar to the CALENDAR intrinsic. The bits and their meanings are as follows: Bits (0:16) Unused Bits (16:7) The year of the century Bits (23:9) The day of the year This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3085	**Password Expiration Warning (I32) Put: No; Verify: Yes; Release 5.0** Returns the warning interval in days for all user passwords that are set to expire on the next global expiration date. A valid range of values is 0 to 364 days. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3086	**Embedded Password Disallow (B) Put: No; Verify: Yes; Release 5.0** Returns true when :STREAM command will not accept optional embedded password syntax on the JOB card, regardless of the password validity. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3087	**Cross Stream Restriction and Authorization (I32) Put: No; Verify: Yes; Release 5.0** Returns a value of one when the Cross Stream Restriction is enabled. This prevents a person without SM or AM from streaming jobs that log on another user. A value of three is returned when the Cross Streaming Authorization supplemental option is enabled. This allows the creator to stream a "protected job", when the creator name and user name in the job card match. The bits and their meanings follow: Bits (0:30) Unused Bits (30:1) Cross Stream Authorization Bits (31:1) Cross Stream Restriction This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3088	**Stream Privilege and Authorization (I32) Put: No; Verify: Yes; Release 5.0** Returns a value of one when the Stream Privilege feature is enabled. This allows certain authorized users to stream particular jobs with no password verification requirement. This privilege is granted to users with SM, or AM capabilities within an account, and the job's owner. A value of three is returned when the privilege authorization mechanism is enabled. This means the stream privilege can also be extended to the creator, when the creator name and user name in the job card match. Bits (0:30) Unused Bits (30:1) Stream Privilege Authorization Bits (31:1) Stream Privilege This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3089	**Assurance of Auditability (B) Put: No; Verify: Yes; Release 5.0** Returns true if the assurance of auditability feature is set. When enabled, all jobs and sessions will log off if any system logging errors occur. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3090	**Maximum File Protection (B) Put: No; Verify: Yes; Release 5.0** Returns true when maximum file protection is enabled. When enabled, if no ACD is attached to a newly created file, a system default ACD is automatically defined for the file. After the file is created with the default ACD, users other than the CREATOR of the file, or users with SM or AM capability will be denied file access. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3091	**Global User Password Maximum Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the maximum period in days for which a password is valid; this includes the expiration period. Password aging is enforced only on REQUIRED user passwords. A valid range of values is 0 to 365 days. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3092	**Global User Password Minimum Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the minimum period in days a new or changed user password cannot be altered. Password aging is enforced only on REQUIRED user passwords. A valid range of values is 0 to 364 days. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3093	**Global User Password Expiration Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the number of days a user password is expired before becoming invalid. A user can still change an expired password. Once the password exceeds the expired period it is placed in an invalid state. Once invalid, only the system or account manager can change the password. Password aging is enforced only on REQUIRED user passwords. A valid range of values is 0 to 364 days. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3094	**Global User Password Warning Days (I32) Put: No; Verify: Yes; Release 5.0** Returns the warning period in days the user is given before the password expires. Password aging is enforced only on REQUIRED user passwords. A valid range of values is 0 to 364 days. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.
3095	**Maximum Invalid User Logons (I32) Put: No; Verify: Yes; Release 5.0** Returns the maximum number of invalid user logon attempts before the user becomes invalid. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide*.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3096	**Disabled User Timeout (B) Put: No; Verify: Yes; Release 5.0** Returns the user ID timeout value in seconds. When the timeout expires for the invalid user the Security process will enable the user. Changing this value does not impact any user timeouts currently in effect. The valid range of values is 0 to 32766 seconds. This feature is set in the Global Security Options Menu of the HP Security Monitor. For more information see the *MPE/iX Security Features System Manager's Guide.*
3097	**Security Installed (B) Put: No; Verify: Yes; Release 5.0** Returns true when the HP Security Monitor is installed. For more information see the *MPE/iX Security Features System Manager's Guide.*
3099	**Number of currently configured workgroups (I32) Put: No; Verify: Yes** Returns the number of workgroups on the system. The minimum value that would be returned is 5, which represents the five default workgroups.
3100	**Purge Scan (B) Put:Yes; Verify: Yes** This item for AIFSCGET returns the value denoting that a scan of all processes that belong to purged workgroups is in progress or not. However for AIFSCPUT if this item is passed, it is used as an option to explicitly initiate a purge-pending scan. A purge pending scan checks processes that are assigned to purge-pending workgroups for reassignment to other workgroups.
3101	**System-wide Scan pending? (B) Put:No; Verify: Yes** Returns or verifies value denoting that a scan of all processes for reassignment to workgroups is pending or not.
3102	**Lower job number limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the lower limit of the range of job numbers to be assigned. Job numbers will be assigned outside of this range only when all job numbers within the range are in use. The lower limit must be a positive integer less than the current upper limit or the absolute job number limit (16383), whichever is less. An upper limit of 0 is considered the same as the absolute job number limit.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3103	**Upper job number limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the upper limit of the range of job numbers to be assigned. Job numbers will be assigned outside of this range only when all job numbers within the range are in use. The upper limit must either be 0, or a positive integer greater than the current lower limit but less than or equal to the absolute job number limit (16383). An upper limit of 0 is considered the same as the absolute job number limit.
3104	**Lower session number limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the lower limit of the range of session numbers to be assigned. Session numbers will be assigned outside of this range only when all session numbers within the range are in use. The lower limit must be a positive integer less than the current upper limit or the absolute session number limit (16383), whichever is less. An upper limit of 0 is considered the same as the absolute session number limit.
3105	**Upper session number limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the upper limit of the range of session numbers to be assigned. Session numbers will be assigned outside of this range only when all session numbers within the range are in use. The upper limit must either be 0, or a positive integer greater than the current lower limit but less than or equal to the absolute session number limit (16383). An upper limit of 0 is considered the same as the absolute session number limit.
3106	**Lower input spoolid limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the lower limit of the range of input spoolids to be assigned. Input spoolids will be assigned outside of this range only when all input spoolids within the range are in use. The lower limit must be a positive integer less than the current upper limit or the absolute input spoolid limit (9999999), whichever is less. An upper limit of 0 is considered the same as the absolute input spoolid limit.
3107	**Next input spoolid (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the number assigned to the next input spoolid, value in the range 1..9999999. Do not set this value to that of an existing input spoolid. You may set a value outside the range defined by the lower and upper input spoolid limits, but it will not be used unless the limits are changed such that the value is in range before the system needs to assign it to a spoolid. This is because the system always tries to assign a value within limits. See the `SETCOUNTER` command description for further details.
3108	**Upper input spoolid limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the upper limit of the range of input spoolids to be assigned. Input spoolids will be assigned outside of this range only when all input spoolids within the range are in use. The upper limit must either be 0, or a positive integer greater than the current lower limit but less than or equal to the absolute input spoolid limit (9999999). An upper limit of 0 is considered the same as the absolute input spoolid limit.

Table 3-30. System Configuration Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Description
3109	**Lower output spoolid limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the lower limit of the range of output spoolids to be assigned. Output spoolids will be assigned outside of this range only when all output spoolids within the range are in use. The lower limit must be a positive integer less than the current upper limit or the absolute output spoolid limit (9999999), whichever is less. An upper limit of 0 is considered the same as the absolute output spoolid limit.
3110	**Next output spoolid (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the number assigned to the next output spoolid, a value in the range 1..9999999. Do not set this value to that of an existing output spoolid. You may set a value outside the range defined by the lower and upper output spoolid limits, but it will not be used unless the limits are changed such that the value is in range before the system needs to assign it to a spoolid. This is because the system always tries to assign a value within limits. See the SETCOUNTER command description for further details.
3111	**Upper output spoolid limit (I32) Put: Yes; Verify: Yes; Release 5.0** Returns or modifies the upper limit of the range of output spoolids to be assigned. Output spoolids will be assigned outside of this range only when all output spoolids within the range are in use. The upper limit must either be 0, or a positive integer greater than the current lower limit but less than or equal to the absolute output spoolid limit (9999999). An upper limit of 0 is considered the same as the absolute output spoolid limit.
3112	**Workgroup Creation Count (I32) Put: No; Verify: Yes** Returns the number of times the workgroup file has been regenerated. Everytime a change is made to a workgroup or the set of workgroups, this count is incremented. Thus it can be used to verify whether the workgroup set has been modified between AIF calls.

AIFSPFGET

Returns spool file information.

Syntax

```
                    REC              I32A           @64A           RECA
AIFSPFGET (overall_status, itemnum_array, item_array, itemstatus_array,
                   @64     REC      I32
              spf_addr, spf_id, user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

spf_addr **64-bit address by value (optional)**

Passes the virtual address of the spool file for which information is desired.

Default: nil

spf_id **record by reference (required)**

Passes the spool file ID of the spool file for which information is desired. If *spf_addr* also passed, ***spf_id*** is used for validation.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

AIFSPFGET accepts as an input key a spool file ID and/or a spool file address, which identifies the spool file that the information is to be returned on. The spool file address is the faster mechanism of the two. If both are provided, the address and the spool file ID are checked against each other to make sure that they match the same spool file. If they don't match, the spool file ID will be used as it is the unique key while the spool file address is reusable and not unique. The mismatch of the two keys indicates that the spool file entry pointed to by the address is no longer used by the file identified by the spool file ID. It is most likely that the spool file has been purged and the entry has been reused by a later created spool file. AIFSPFGET will then use the spool file ID to verify the existence of the spool file.

If the two keys match, the spool file address will be used. Note that the address is not a unique identifier. The address may be pointing to a deallocated entry where the information of a deleted spool file is still available. Therefore, it is possible for AIFSPFGET to successfully return information for a spool file that has been deleted but the spool file address and ID do not match.

AIFSPFGET Items

The following two tables provide summary and detailed descriptions of the item numbers associated with spool file information.

Item Summary

The following table summarizes the item numbers associated with spool file information. For more detailed information about these item numbers, refer to the tables of `AIFSPFGET` and `AIFSPFPUT` item descriptions.

Table 3-31. Spool File Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
8501	I32	File state	Y	Y	0	10	-8043
8502	I32	Priority	Y	Y	0	14	-8047
8503	I32	Restartable?	N	Y			
8504	I32	Disposition	Y	Y	1	2	-8048
8505	I32	Private?	N	Y			
8506	I32	Forms message?	N	Y			
8507	I32	Incomplete?	Y	Y	0	1	-8049
8508	I32	Job or data file?	N	Y			
8509	I32	Aborted `$STDLIST`?	Y	Y	0	0	0
8510	I32	Spool file ID	N	Y			
8511	I32	Copies requested	Y	Y	1	65535	-8050
8512	I32	Ready date	Y	Y			
8513	I32	Ready time	Y	Y			
8514	I32	Number of pages	Y	Y			
8515	I32	Restart page	Y	Y			
8516	CA32	Creator name/account	Y	Y			
8517	I32	Job/session number	Y	Y			
8518	CA16	Job name	Y	Y			
8519	CA16	Spool file designator	Y	Y			
8520	CA18	Target device	Y	Y			
8521	I32	Device record size	N	Y			
8522	BIT16	Device type	N	Y			
8523	BIT16	Device subtype	N	Y			
8524	I32	Completed copy count	Y	Y	1	65535	-8050
8525	CA16	Forms ID	Y	Y			
8526	UFID_type	Spool file UFID	N	Y			
8527	CA18	Active device	N	Y			
8528	I32	Number of records	N	Y			
8529	I32	Number of sectors	N	Y			
8530	CA32	Environment file name	N	Y			
8531	BIT16	Foptions	N	Y			
8532	BIT16	Aoptions	N	Y			
8533	I32	File open flag	N	Y			
8533	I32	Broadcastable	Y	Y			

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with spool file information returned by AIFSPFGET.

Table 3-32. AIFSPFGET Spool File Information Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
8501	**File state (I32) Release 3.0** Returns the state of the spool file. Values and their meanings are as follows: 0 Open state (job/data input spool file being accessed) 1 Active state (job/data input spool file being created) 2 Create state (output spool file being created) 3 Defer state (defer option specified for output spool file) 4 Ready state (spool file ready to be input or output) 5 Transfer state (output spool file being transferred to remote node) 6 Print state (output spool file being printed on a device) 7 Problem state (abnormality preventing output spool file from printing) 8 Del_pending state (output spool file to be deleted after closing) 9 Spsave state (output spool file copies printed, SPSAVE option specified) 10 (Reserved)
8502	**Priority (I32) Release 3.0** Returns the output priority of the spool file, a value in the range 0..14. Valid only for output spool files.
8503	**Restartable? (I32) Release 3.0** Returns whether or not the job is restartable. RESTART is a parameter of the JOB command. This item applies only to the $STDIN of a job input spool file. Values and their meanings are as follows: 0 Not a restartable job 1 Restartable job
8504	**Disposition (I32) Release 3.0** Returns whether the spool file is to be saved or purged after it has been printed. Values and their meanings are as follows: 1 Save after printing 2 Purge after printing
8505	**Private? (I32) Release 3.0** Returns whether the spool file is a private spool file. All input spool files are created with the private option. Values and their meanings are as follows: 0 Not a private spool file 1 Private spool file

Table 3-32. AIFSPFGET Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
8506	**Forms message? (I32) Release 3.0** Returns whether the spool file has a forms message associated with it. Valid only for output spool files. Values and their meanings are as follows: 0 No forms message associated 1 Yes forms message associated
8507	**Incomplete? (I32) Release 3.0** Returns whether the spool file is incomplete. Valid only for output spool files. Values and their meanings are as follows: 0 Complete 1 Incomplete
8508	**Job or data file? (I32) Release 3.0** Returns whether the spool file is a job or data file. Values and their meanings are as follows: 0 Neither a job file or a data file (for output spool file) 1 Job file 2 Data file
8509	**STDLIST of aborted job? (I32) Release 3.0** Returns whether or not the spool file is the `$STDLIST` of an aborted job. Valid only for output spool files. Values and their meanings are as follows: 0 Not `$STDLIST` of an aborted job 1 `$STDLIST` of an aborted job
8510	**Spool file ID (REC) Release 3.0** Returns the spool file ID in the following format: Bits (0:31) The spool file ID number in the form of 31-bit positive integer Bits (31:1) 0 for input and 1 for output spool file Record type: `spf_id_type` (Refer to appendix B.)
8511	**Copies requested (I32) Release 3.0** Returns the total number of copies requested for the spool file. Valid only for output spool files.
8512	**Ready date (REC) Release 3.0** Returns the calendar date when the spool file was created. All fields are 0's if the spool file is not yet `READY` or not out of the `ACTIVE/CREATE` state. The format returned in the 32-bit integer is the same as that returned by the `CALENDAR` intrinsic. The bits and their meanings are as follows: Bits (0:16) Unused Bits (16:7) The year of the century Bits (23:9) The day of the year Record type: `bit32` (Refer to appendix B.)

Table 3-32. AIFSPFGET Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
8513	**Ready time (REC) Release 3.0** Returns the time (hours/minutes/seconds/tenths of seconds) when the spool file was created. All fields are 0's if the spool file is not yet ready or is not out of the active/create state. The format returned in the 32-bit integer is the same as that returned by the `CLOCK` intrinsic. The bits and their meanings are as follows: Bits (0:8) The hour of the day Bits (8:8) The minute of the hour Bits (16:8) The seconds Bits (24:8) The tenths of seconds Record type: `bit32` (Refer to appendix B.)
8514	**Number of pages (I32) Release 3.0** Returns the number of pages in the spool file. A positive number indicates the actual number of pages in the spool file. A negative number indicates that the spool file has never been printed before, and the number is only an estimation. Valid only for output spool files.
8515	**Restart page (I32) Release 3.0** Returns the page at which to restart if the printing of the spool file has been suspended. Valid only for output spool files.
8516	**User name and account name of creator (CA32) Release 3.0** Returns the user and account name of the creator of the spool file. The first 16 bytes is the user name, and the second 16 bytes is the account name. The names should be left-justified and padded with blanks. Currently, only the first 8 bytes of each field are used.
8517	**Job/session number (REC) Release 3.0** Returns the job/session number under which the spool file was created. A job/session number with a single quote (') indicates that the job or session is not current to the system, but rather that of a spool file . The format of the data returned is: Bits (0:2) Job or session (see below)? Bits (2:30) The job/session number The value of the first two bits indicates: 0 Session not current to the system (recovered) 1 Session current to the system 2 Job current to the system 3 Job not current to the system (recovered) Record type: `bit32` (Refer to appendix B.)
8518	**Job name (CA16) Release 3.0** Returns the job name under which the spool file was created.

Table 3-32. AIFSPFGET Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
8519	**File designator (CA16) Release 3.0** Returns the formal file designator of the spool file.
8520	**Target device (REC) Release 3.0** Returns the destination device name or device class for the spool file. Record type: `device_name_type` (Refer to appendix B.)
8521	**Device record size (REC) Release 3.0** Returns the record size of the target device of the spool file. Record type: `bit32` (Refer to appendix B.)
8522	**Device type (REC) Release 3.0** Returns the device type of the target device of the spool file. Record type: `bit16` (Refer to appendix B.)
8523	**Device subtype (REC) Release 3.0** Returns the device subtype of the target device of the spool file. Record type: `bit16` (Refer to appendix B.)
8524	**Completed copy count (I32) Release 3.0** Returns the number of copy that has already been printed. Valid only for output spool files.
8525	**Forms ID (CA16) Release 3.0** Returns the forms ID of the spool file. Valid only for output spool files.
8526	**Spool file UFID (REC) Release 3.0** Returns the UFID of the spool file. Record type: `ufid_type` (Refer to appendix B.)
8527	**Active device (REC) Release 3.0** Returns the name of the last device that has picked up a copy of the spool file and is currently processing the spool file. Valid only for output spool files. Record type: `device_name_type` (Refer to appendix B.)
8528	**Number of records (REC) Release 3.0** Returns the number of records in the spool file. However, this number is only valid after the spool file has been created successfully. Record type: `bit32` (Refer to appendix B.)

Table 3-32. AIFSPFGET Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
8529	**Number of sectors (I32) Release 3.0** Returns the number of sectors in the spool file.
8530	**Environment File Name (CA36) Release 3.0** Returns the environment file name of the spool file. This field is valid only if the system has not been rebooted since the file was created. Valid only for output spool files.
8531	**Foptions (REC) Release 3.0** Returns the Foptions of the spool file. This field is valid only if the system has not been rebooted since the file was created. Record type: `bit16` (Refer to appendix B.)
8532	**Aoptions (REC) Release 3.0** Returns the Aoptions of the spool file. This field is valid only if the system has not been rebooted since the file was created. Record type: `bit16` (Refer to appendix B.)
8533	**File Open Flag (I32) Release 3.0** Returns whether the spool file is currently opened by `HPFOPEN` or `FOPEN` and whether the file is opened or exclusively opened. The values and their meanings are as follows: 0 The file is not open 1 The file is opened exclusively 2 The file is opened
8534	**Broadcastable (I32) Release 4.0** Returns whether the broadcastable field of an output spoolfile is set. Used to print additional copies of SPSAVE'd output spool files. Valid only for output spool files.

AIFSPFLINK

Programmatically executes the MPE/iX command `SPOOLF` *spool_id*;`PRINT`.

Syntax

```
                    REC          REC          REC          REC
AIFSPFLINK (overall_status, source_spf, linked_spf_id, linked_spf_ufid,
                   REC       I32     I32     I32     I32
              target_device, priority, copies, spsave, defer,
                    CA       I32
              spf_lockword, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

source_spf **record by reference (required)**

Passes the name of the spool file to be copied and linked.

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

linked_spf_id **record by reference (optional)**

Returns the spool file ID of the spool file created and linked to the `HPSPOOL` account.

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

Default: nil

linked_spf_ufid **record by reference (optional)**

Returns the UFID of the spool file created and linked to the `HPSPOOL` account.

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

Default: nil

target_device **Record by reference (optional)**

Passes the device name used as the new target device for printing the spool file. Whether this parameter is specified or not, the spool file queue for the device must be open, or an error results. The device name should be left-justified and padded with blanks.

Array type: `device_name_type` (Refer to appendix B.)

Default: nil

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

Passes the output priority of the newly created spool file in the `HPSPOOL` account. The valid range is 0..13.

Default: 8

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

Passes the number of copies to be printed for the newly created spool file in the `HPSPOOL` account. The valid range is 1..65535.

Default: 1

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

Passes the SPSAVE flag setting for the newly created spool file in the `HPSPOOL` account. The `SPSAVE` flag directs the spooler to save the spool file in the `HPSPOOL` account after it has been printed. The default is not to save the spool file. The values are as follows:

0 No `SPSAVE`
1 Yes `SPSAVE`

Default: 0

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

Passes the file state of newly created spool file. If defer is specified, the spool file is not printed. The default is not to defer the printing of the spool file. The values are as follows:

0 No defer
1 Yes defer

Default: 0

spf_lockword **character array by reference (optional)**

Passes the lockword for spool file specified in ***source_spf***.

Array type: `pac16` (Refer to appendix B.)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

Spool files under the native mode spooler are permanent disk files. They are created implicitly by sending data to a spooled output device. Spool files can be created explicitly by using the `HPFOPEN` intrinsic or the `BUILD` command.

A spool file created by `HPFOPEN` with the link option or the linked device option resides in the `HPSPOOL` account. It has an entry in the separately maintained spool file directory and is linked into the spool file queues. This is called a "linked" spool file and it is known to the spooling processes.

A spool file created by the `BUILD` command or by the `HPFOPEN` intrinsic without the link option can reside in any user directory. It does not have an entry in the spool file directory and is not linked into the spool file queues. A spool file created in such a manner is described as "unlinked" and is not known to the spooling processes. To clarify this further, A linked spool file must reside in the `HPSPOOL` account, but a spool file that resides in the `HPSPOOL` account is not necessarily linked.

To link a spool file for printing, the spool file must first be copied to the HPSPOOL account and linked into the spool file directory. AIFSPFLINK provides both the copying and the linking as described. AIFSPFLINK works for both a linked or an unlinked spool file, but if a spool file is already linked, it is not necessary to call AIFSPFLINK to get extra copies of the spool file printed.

One other application of AIFSPFLINK is that a user may save a copy of a spool file from the HPSPOOL account in his own directory. This can be achieved by the COPY command. The user copy of the spool file is not linked. Later on, when this spool file is to be printed, the user can call AIFSPFLINK to copy and link the spool file to the HPSPOOL account. However, the spool file queue must be open for the target device before a copy of the spool file can be created in the OUT group of the HPSPOOL account.

Certain attributes of a spool file can be altered while linking the spool file by calling this routine. If the target device information exists in the file label extension, then that device will be used as the default. The *target_device* parameter may be specified to override the existing target device. If there is no target device in the file label extension, *target_device* must be specified when calling AIFSPFLINK or an error results.

If a lockword exists for the file specified in *source_spf*, it must be specified in the parameter *spf_lockword*. Other attributes that can be changed for the spool files are priority, copies, spsave and defer.

AIFSPFLIST

Returns the virtual address and the spool file IDs of spool files that meet the specified selection criteria.

Syntax

```
                    REC        REC       @64A          RECA
AIFSPFLIST (overall_status, seleq, spf_addr_array, spf_id_array,
                 I32      I32         B
            spf_count, user_id, stop_search);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

seleq **record by reference (optional)**

Passes the selection criteria to be used as the filter for selecting the spool file to be returned by this routine. The maximum length of any selection equation is 277 characters. If defaulted to nil, all spool files on the system are returned.

The actual parameter should be delimited by the beginning and ending square brackets, "[" and "]". Blanks count as characters.

For example, to list all spool files owned by the user `JON.DOE` that are in the `READY` state, pass the following in the *seleq* parameter:

```
[(OWNER = JON.DOE) AND (STATE = READY)]
```

The character string passed is the same as that passed in the `LISTSPF` command.

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

Default: nil

spf_addr_array **64-bit address array by reference (optional)**

Returns virtual addresses specific to the spool files qualifying the selection criteria. If more qualifying spool files are found than can be returned in this array, it is only filled to its maximum capacity. However, the total number of qualifying spool files found is returned in *spf_count*.

Array type: `globalanyptr`

Default: nil

spf_id_array **record array by reference (optional)**

Returns the file IDs of the spool files selected. If more qualifying spool files are found than can be returned in this array, it is only filled to its maximum capacity. However, the total number of qualifying spool files is returned in *spf_count*.

Array type: array of `spf_id_type` (Refer to appendix B.)

Default: nil

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

Passes the number of elements available in the *spf_addr_array* and the *spf_id_array*.

Upon return, *spf_count* holds the number of spool files that qualify the selection equation.

If the arrays are too small to hold all the qualifying spool files, *spf_count* returns the total number of qualifying spool files instead of the number of entries returned in the arrays. A warning is also returned in the ***overall_status*** parameter.

If *stop_search* is not passed or set to false, the total number of qualified spool files on the system is returned. If *stop_search* is set to true, only up to user-specified *spf_count* of qualified spool files will be returned.

Caution

Since *spf_addr_array* and *spf_id_array* are passed by Pascal uncheckable anyvar, there is no way for this routine to tell the actual size of the arrays. It is important that you initialize this parameter to the number of elements in the *spf_addr_array* and/or *spf_id_array* before calling this routine.

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

stop_search **Boolean by value (optional)**

If true, search will stop after *spf_count* number of qualified spool files have been found. Default is searching for all the qualified spool files on the system.

Default: False

Operation Notes

The selection criteria is specified in the *seleq* parameter. The number of the spool files qualifying the selection criteria is returned in the *spf_count* parameter.

The spool file queues are organized by device class and device name. Within each queue, the spool files are sorted by priority and ready date/time. This procedure scans the spool file queue and finds all spool files satisfying the specified selection criteria. The spool files returned in the arrays will be in the same sequence as listed by the `LISTSPF` command, for example, sorted by device, priority and ready date/time.

Spool file address's are returned in the *spf_addr_array* and spool file IDs are returned in the *spf_id_array*. Either or both arrays can be defaulted to nil; in this case, no spool file ID or address is returned.

The list of spool file attributes that can be selected on are as follows:

- Device name
- File designator
- Spool file ID
- Number of pages in the spool file
- Form ID
- Spool file state
- Job name
- Disposition (`SPSAVE` or `PURGE`)
- Number of copies
- Priority
- Job number
- Number of records in the spool file
- Owner of the spool file
- `$STDLIST` of aborted job
- Spool file ready date

AIFSPFPUT

Modifies spool file information.

Syntax

```
                    REC              I32A            @64A             RECA
AIFSPFPUT (overall_status, itemnum_array, item_array, itemstatus_array,
               @64      REC      I32         I32A           @64A
           spf_addr, spf_id, user_id, ver_item_nums, ver_items,
                    RECA
           ver_item_statuses) ;
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

spf_addr **64-bit address by value (optional)**

Passes the virtual address of the spool file for which information is to be modified.

Default: nil

spf_id **record by reference (required)**

Passes the ID of the spool file for which information is to be modified. If *spf_addr* is also passed, ***spf_id*** is used for validation.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element $n+1$ must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

Operation Notes

AIFSPFPUT accepts a spool file ID or spool file address that identifies the spool file and modifies the information in the spool file directory and file label extension. If both the address and the spool file ID are provided, they are checked against each other to make sure that they match the same file. If they don't match, the spool file ID will be used as it is the unique key while the spool file address is reusable and not unique. The mismatch of the two keys indicates that the spool file entry pointed to by the address is no longer used by the file identified by the spool file ID. It is most likely that the spool file has been purged and the entry has been reused by a later created spool file. AIFSPFGET will then use the spool file ID to verify the existence of the spool file.

If the two keys match, the spool file address will be used. Note that the address is not a unique identifier. The address may be pointing to a deallocated entry where the information of a deleted spool file is still available. Therefore, it is possible for AIFSPFGET to successfully return information for a spool file that has been deleted but the spool file address and ID do not match.

The three verification arrays are used to verify that the spool file entry to be updated in the SPFDIR is in the expected state. If the three verification arrays are passed, all item values specified by them are checked against the current values. The modification is performed only if all values match. The three verification arrays must all be passed or all defaulted when calling AIFSPFPUT, otherwise, an error is returned.

Caution

All ASCII characters to be written must be upshifted before calling AIFSPFPUT.

AIFSPFPUT Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with spool file information modified by AIFSPFPUT.

Table 3-33. AIFSPFPUT Spool File Information Item Descriptions

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
8501	**File state (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the state of the spool file. Values and their meanings are as follows: 0 Open state (job/data input spool file being accessed) 1 Active state (job/data input spool file being created) 2 Create state (output spool file being created) 3 Defer state (defer option specified for output spool file) 4 Ready state (spool file ready to be input or output) 5 Transfer state (output spool file being transferred to remote node) 6 Print state (output spool file being printed on a device) 7 Problem state (abnormality preventing output spool file from printing) 8 Del_pending state (output spool file to be deleted after closing) 9 Spsave state (output spool file copies printed, SPSAVE option specified) 10 (Reserved) If a spool file is in the create state, the only legal state change is to defer or del_pending. This operation sets only an internal defer or delete pending flag. The change of file state to defer or del_pending occurs only when the file has been successfully created. If the original state of the spool file is del_pending, changing the state of the spool file only changes the value of this field. The internal delete pending flag is not turned off. Changing a currently printing spool file state to del_pending or defer causes the spooler to terminate the print job immediately. Changing the spool file state to del_pending flags that the spool file is to be deleted the next time the spool file is closed. In order to have the file actually deleted, you must FOPEN and FCLOSE the file if the file was not already opened when the state was changed. Do not change the state of an output spool file to create, open, or active. Do not change the state of a spool file to spsave if the spool file is private. Do not change the state of an input spool file, except for changing the state of an data input spool file from ready to del_pending.
8502	**Priority (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the output priority of the spool file, a value in the range 0..14. Changing the priority of a spool file causes the file to be requeued in the SPFDIR. If the priority is higher than the outfence of the target device, the spool file is printed. Valid only for output spool files.

Table 3-33. AIFSPFPUT Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
8504	**Disposition (I32) Put: Yes; Verify: Yes; Release 3.0** Indicates whether the spool file is to be saved or purged after it has been printed. It is illegal to change the disposition of a private spool file to save. Values and their meanings are as follows: 1 Save 2 Purge
8507	**Incomplete? (I32) Put: Yes; Verify: Yes; Release 3.0** Indicates whether the spool file is incomplete by setting the "incomplete due to lack of space" flag for the specified spool file. Valid only for output spool files. Values and their meanings are as follows: 0 Complete 1 Incomplete
8509	**STDLIST of aborted job? (I32) Put: Yes; Verify: Yes; Release 3.0** Indicates whether the spool file is the `$STDLIST` of an aborted job by modifying the "`$STDLIST` of aborted job" flag. Valid only for output spool files. Values and their meanings are as follows: 0 Not `$STDLIST` of an aborted job 1 `$STDLIST` of an aborted job
8511	**Copies requested (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the total number of copies requested for the spool file. Valid only for output spool files. Do not modify this field if the spool file is private. Changing the number of copies to greater than the copies printed causes a ready spool file to be message sent to the spooler of the target device. In this case, the device is activated if it is in the idle state. Changing the number of copies to less than or equal to the number of completed copies (refer to item 8524) might cause the spool file to stay on the system even if the file has not been marked as spsave. Caution should be exercised when changing this field.
8512	**Ready date (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the calendar date when the spool file was created. This field cannot be modified if the file is in the create or active state. The format of the 32-bit integer passed is the same as that returned by the `CALENDAR` intrinsic. The bits and their meanings are as follows: Bits (0:16) Unused (must be set to zeros) Bits (16:7) The year of the century Bits (23:9) The day of the year Record type: `bit32` (Refer to appendix B.)

Table 3-33. AIFSPFPUT Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
8513	**Ready time (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the time (hours/minutes/seconds/tenths of seconds) when the spool file was created.This item cannot be modified if the file is in the create or active state. The format passed in the 32-bit integer is the same as that returned by the `CLOCK` intrinsic. The bits and their meanings are as follows: Bits (0:8) The hour of the day Bits (8:8) The minute of the hour Bits (16:8) The seconds Bits (24:8) The tenths of seconds Record type: `bit32` (Refer to appendix B.)
8514	**Number of pages (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of pages in the spool file. A positive number indicates the actual number of pages in the spool file. A negative number indicates that the spool file has never been printed before and the number is only an estimation. Valid only for output spool files.
8515	**Restart page (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the page at which to restart if the printing of the spool file has been suspended. Negative values are not valid for this field. Valid only for output spool files.
8516	**User name and account name of creator (CA32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the user name and account name of the creator of the spool file. The first 16 bytes is the user name, and the second 16 bytes is the account name. The names should be upshifted, left-justified, and padded with blanks. Currently, only the first 8 bytes of each field is used. Valid only for output spool files.
8517	**Job/session number (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the job/session number under which the spool file was created. The format of the data returned is as follows: Bits (0:2) Job or session (see below)? Bits (2:30) The job/session number The value of the first two bits indicates the following: 0 Session not current to the system (recovered) 1 Session current to the system 2 Job current to the system 3 Job not current to the system (recovered) Record type: `bit32` (Refer to appendix B.)
8518	**Job name (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the job name under which the spool file was created. The name should be upshifted, left-justified and padded with blanks. Valid only for output spool files.

Table 3-33. AIFSPFPUT Spool File Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Put; Verify; Release First Available Description
8519	**Spool file designator (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the formal file designator of the spool file. The file designator should be upshifted, left-justified, and padded with blanks.
8520	**Target device (REC) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the destination logical device for the spool file. It can be a device name, a device number or a device class. A device number must be in the format of ASCII character string. Changing the target device causes the spool file entry to be moved to the new target device's queue. Only users with SM capability can change the target device of a private spool file. The device name must be upshifted, left-justified, and padded with blanks. Valid only for output spool files. Record type: `device_name_type` (Refer to appendix B.)
8524	**Completed copy count (I32) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the number of copies that have been printed already. Valid only for output spool files.
8525	**Forms ID (CA16) Put: Yes; Verify: Yes; Release 3.0** Returns or modifies the forms ID of the spool file. The forms ID must be upshifted, left-justified, and padded with blanks. Valid only for output spool files.
8533	**Broadcastable (I32) Put:Yes; Verify: Yes; Release 4.0** Returns or modifies the broadcastable field of the output spoolfile. Used to print additional copies of SPSAVE'd output spool files by modifying copies state field along with broadcastable field for output spool files. Valid only for output spool files.

AIFSPPGET

Returns spooler process information.

Syntax

```
                  REC             I32A          @64A          RECA
AIFSPPGET (overall_status, itemnum_array, item_array, itemstatus_array,
                  REC         I32
           spooler_device, user_id);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the information to be returned to a data structure pointed to in the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

spooler_device **record by reference (required)**

Passes the device name or logical device number (LDEV) of the device owned by a spooler process for which information is desired. A logical device number (LDEV) must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

When the spooler on a device is stopped, the spooler process entry will be deallocated and no longer available for any access. Any reference to that spooler process will return error -8003: The device is not spooled.

AIFSPPGET Items

The following two tables provide summary and detailed descriptions of the item numbers associated with spooler process information.

Item Summary

The following table summarizes the item numbers associated with spooler process information. For more detailed information about these item numbers, refer to the tables of `AIFSPPGET` and `AIFSPPPUT` spooler process information item descriptions.

Table 3-34. Spooler Process Information Item Summary

Item	Type	Description	Put	Ver	Min	Max	Error#
8001	I32	LDEV Number	N	Y			
8002	I32	Process PIN	N	Y			
8003	I32	Current spool file ID	N	Y			
8004	I32	Process kind	N	Y			
8005	I32	Process state	N	Y			
8006	I32	Finish strategy	N	Y			
8009	I32	Device outfence	Y	Y	0	14	
8010	I32	Suspend keep flag	N	Y			

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with spooler process information returned by `AIFSPPGET`.

Table 3-35. AIFSPPGET Spooler Process Information Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
8001	**LDEV number (I32) Release 3.0** Returns the LDEV number of the specified device.
8002	**Process PIN (I32) Release 3.0** Returns the process identification number (PIN) of the spooler process for the specified device.
8003	**Current spool file ID (REC) Release 3.0** Returns the spool file ID of the spool file that is currently being printed on the device. Record type: `spf_id_type` (Refer to appendix B.)
8004	**Process kind (I32) Release 3.0** Returns the process kind of the spooler. Values and their meanings are as follows: 0 NoSpool 1 InSpool 2 OutSpool 3 (reserved) 4 (reserved)
8005	**Process state (I32) Release 3.0** Returns the state of the spooler process. Values and their corresponding states are as follows: 0 Initialization 1 Release 2 Start 3 Stop 4 Stop_Pending 5 Suspend 6 Suspend_Pending 7 Resume 8 Active 9 Shut_Pending 10 Idle

Table 3-35. AIFSPPGET Spooler Process Information Item Descriptions (continued)

Item Number	Item Name (Data Type) Release First Available Description
8006	**Finishing strategy (I32) Release 3.0** Returns the finishing strategy of the device. This is one of the options that can be specified for the `SPOOLER`*nn*`;STOP/SUSPEND` command. It is only valid when the spooler process is being suspended or stopped. Values and their meanings are as follows: 0 None (device not being suspended/stopped) 1 Now 2 End of copy
8009	**Device outfence (I32) Release 3.0** Returns the current outfence of the device, a value in the range 0..14.
8010	**Suspend keep flag (I32) Release 3.0** Returns a value indicating whether the spooler is to retain ownership of the spool file or to close the spool file and return it to the `READY` state. This is one of the options that can be specified for the `SPOOLER`*nn*`; SUSPEND` command. The field is valid only when the spooler process is being suspended. Values and their meanings are as follows: 0 None (suspend not currently in effect) 1 Keep 2 No keep

AIFSPPOPENQ

Opens the spool queue for the specified logical device number (LDEV), device name, or device class.

Syntax

```
                    REC           REC          I32
AIFSPPOPENQ (overall_status, spooler_device, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

The logical device number (LDEV), device name, or device class for which a spool queue is to be opened. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFSPPOPENQ` is the programmatic interface for executing the `OPENQ` command.

AIFSPPPUT

Modifies spooler process information.

Syntax

```
                    REC            I32A           @64A           RECA
AIFSPPPUT (overall_status, itemnum_array, item_array, itemstatus_array,
                    REC          I32        I32A          @64A
           spooler_device, user_id, ver_item_nums, ver_items,
                    RECA
           ver_item_statuses)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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 an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in ***item_array***. If n item numbers are being requested, element $n+1$ must be a zero to indicate the end of the element list.

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

An array where each element is a 64-bit address pointing to a data structure containing new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in ***itemnum_array***.

Array type: `globalanyptr`

itemstatus_array **record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in ***item_array***. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

spooler_device **record by reference (required)**

Passes the logical device number (LDEV) or device name owned by a spooler process for which information is to be modified. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in *ver_items*. If n items are being verified, element $n+1$ must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in *ver_item_nums*.

Array type: `globalanyptr`

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in *ver_items*. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning. Refer to appendix A for meanings of status values.

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

Default: nil

Operation Notes

`AIFSPPPUT` accepts a device name or logical device number (LDEV) that identifies the spooler process for which information in the spooler process information table (SPIT) is to be modified.

The three verification arrays are used to verify that the spooler process to be modified is in the expected state. If the three verification arrays are passed, all item values specified by them are checked against the current values in the SPIT entry. A modification is performed only if all values match. The three verification arrays must all be passed or all defaulted when calling `AIFSPPPUT`, otherwise, an error is returned.

Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with spooler process information modified by `AIFSPPPUT`.

Table 3-36. AIFSPPPUT Spooler Process Information Item Descriptions

Item Number	Item Name (Data Type) Release First Available Description
8009	**Device outfence (I32) Release 3.0** Modifies the outfence of the device. Values are in the range 1..14. An outfence of 0 means that the system global outfence is in effect for this device.

AIFSPPRELEASE

Releases the spool file that is currently kept by the specified suspended spooler.

Syntax

```
                        REC             REC          I32     I32
AIFSPPRELEASE (overall_status, spooler_device, direction, offset,
                  I32      I32
                q_state, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

Passes the logical device number (LDEV), device name, or device class that is currently keeping a spool file due to an earlier `SUSPEND` with the `KEEP` option. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_type` (Refer to appendix B.)

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

Passes a value that tells the spooler how to apply the *offset* parameter. See also the explanation of the *offset* parameter for absolute and relative offsets. The valid inputs and their meanings are as follows:

0	Relative offset specified in the *offset* parameter
1	Absolute offset specified in the *offset* parameter

Default: 0

offset

32-bit signed integer by value (optional)

Passes a value representing a page offset, either absolute or relative, within the spool file. Together with the *direction* parameter, it tells the spooler where to resume when the file is picked up again for printing.

If absolute is specified in *direction*, printing resumes at that page, absolute from the beginning of the file. If relative is specified in *direction*, then depending on whether *offset* is positive or negative, the offset is adjusted either forward or backward relative to the current location, by the number of pages specified.

No matter which combination of offsets is specified, the final location is limited by the bounds of the file. The default for offset is 0. If the printing of a spool file is to be resumed from the beginning of the file, pass absolute for *direction* and 0 for *offset*.

Default: 0

q_state

32-bit signed integer by reference (optional)

Passes a value indicating whether the spooling queue is to be opened or disabled when the spooling process is resumed. The default is not to change the current *q_state* of the spooler process. The valid inputs and their meanings are as follows:

0	No change to the current *q_state* of the spooling process (default)
1	Openq
2	Shutq

Default: 0

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFSPPRELEASE` closes the spool file and returns it to the ready state. An *offset* may be specified to change the resumption point of the spool file the next time it is selected for printing. `AIFSPPRELEASE` is the programmatic interface for executing the command `SPOOLER` *dev*`;RELEASE`.

AIFSPPRESUME

Resumes a suspended spooling process.

Syntax

```
                    REC           REC            I32      I32
AIFSPPRESUME (overall_status, spooler_device, direction, offset,
                  I32     I32
              q_state, user_id)
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

Passes the logical device number (LDEV), device name, or device class for which the spooling process is to be resumed. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_type` (Refer to appendix B.)

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

Passes a value that tells the spooler how to apply the *offset* parameter. See also the explanation of the *offset* parameter for relative and absolute offsets. The valid inputs and their meanings are as follows:

0	Relative offset specified in the *offset* parameter (default)
1	Absolute offset specified in the *offset* parameter

Default: 0

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

Passes a value representing a page offset, either absolute or relative, within the spool file. Together with the *direction* parameter, it tells the spooler where to resume when the file is picked up again for printing.

If absolute is specified in *direction*, printing resumes at that page, absolute from the beginning of the file. If relative is specified in *direction*, then depending on whether *offset* is positive or negative, the offset is adjusted either forward or backward relative to the current location, by the number of pages specified.

No matter which combination of offsets is specified, the final location is limited by the bounds of the file. The default for offset is 0. If the printing of a spool file is to be resumed from the beginning of the file, pass absolute for *direction* and 0 for *offset*.

Default: 0

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

Passes a value that indicates whether the spooling queue is to be opened or disabled when the spooling process is resumed. The default is not to change the current *q_state* of the spooler process. The valid inputs and their meanings are as follows:

0	No change to the current *q_state* of the spooling process (default)
1	Openq
2	Shutq

Default: 0

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

The spooler must be in the suspend state. If the spooler retained a spool file when it was suspended and the spool file was not subsequently released, the *offset* parameter can be specified. If it is not specified, output resumes where it was left off. `AIFSPPRESUME` is the programmatic interface for executing the commands `RESUMESPOOL` and `SPOOLER` *dev*`;RESUME`.

AIFSPPSHUTQ

Closes the spool queue for the specified logical device number, device name, or device class.

Syntax

```
                    REC              REC         I32
AIFSPPSHUTQ (overall_status, spooler_device, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

Passes the device name, LDEV number or device class for which a spool queue is to be closed. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFSPPSHUTQ` is the programmatic interface for executing the `SHUTQ` command.

AIFSPPSTART

Creates and activates a new spooler process to handle spool files destined for the specified logical device number, device name, or device class.

Syntax

```
                    REC              REC          I32     I32
AIFSPPSTART (overall_status, spooler_device, q_state, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

Passes the logical device number (LDEV), device name, or device class for which the spooling process is to be initiated. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_type` (Refer to appendix B.)

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

Passes a value used to indicate whether the spooling queue is to be enabled or disabled when the spooler process is initiated. The default is Openq for starting a spooler process. If Shutq is specified, it prevents users from generating spool files for that device. It does not prevent the user from printing previously generated spool files. The valid inputs and their meanings are as follows:

0	No change to the current *q_state* of the spooling process
1	Openq (default)
2	Shutq

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

An Openq is done by default when `AIFSPPSTART` is invoked, unless the *q_state* parameter specifies Shutq. `AIFSPPSTART` is the programmatic interface for executing the commands `STARTSPOOL` and `SPOOLER` *dev*`;START`.

AIFSPPSTOP

Terminates spooling to the specified logical device number, device name, or device class. The spooling processes associated with the devices are also terminated.

Syntax

```
                  REC              REC        I32     I32      I32
AIFSPPSTOP (overall_status, spooler_device, finish, q_state, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

Passes the device name, LDEV number, or device class for which the spooling process is to be terminated. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_type` (Refer to appendix B.)

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

Passes the finishing strategy for stopping the spooling process. The valid inputs and their meanings are as follows:

1	Finish now (default)
2	Finish at end of copy

Default: 1

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

Passes a value that indicates whether the spooling queue is to remain open or disabled when the spooling process terminates. The default is Shutq for terminating a spooler process. If Openq is specified, it allows users to generate spool files on that device even when the spooling process has been terminated. The valid inputs and their meanings are as follows:

0	No change to the current *q_state* of the spooling process
1	Openq
2	Shutq (default)

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

A Shutq is done by default when `AIFSPPSTOP` is invoked, unless the *q_state* parameter specifies Openq. `AIFSPPSTOP` is the programmatic interface for executing the commands `STOPSPOOL` and `SPOOLER` *dev*`;STOP`.

AIFSPPSUSPEND

Suspends the spooling processes for the specified logical device number, device name, or device class. Associated spooler processes remain alive, but inactive.

Syntax

```
                    REC              REC          I32   I32    I32
AIFSPPSUSPEND (overall_status, spooler_device, finish, keep, direction,
                  I32   I32    I32
                offset, q_state, user_id);
```

Parameters

overall_status **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: `status_type` (Refer to appendix B.)

spooler_device **record by reference (required)**

Passes the logical device number (LDEV), device name, or device class for which the spooling process is to be suspended. An LDEV number must be converted into an ASCII character string before being passed to this routine. The name should be left-justified and padded with blanks.

Array type: `device_name_type` (Refer to appendix B.)

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

Passes the finishing strategy for suspending the spooling process. The valid inputs and their meanings are as follows:

1	Finish now (default)
2	Finish at end of copy

Default: 1

keep

32-bit signed integer by reference (optional)

Passes a value that tells the spooler whether to retain ownership of the currently printing spool file or to close the file and return it to the ready state. The valid inputs and their meanings are as follows:

1 Keep (default)
2 No keep

Do not pass both the finish end of copy and the Keep flags. Also, do not pass finish-end-of-copy and pass a non-zero offset.

Default: 1

direction

32-bit signed integer by value (optional)

Passes a value that tells the spooler how to apply the *offset* parameter. See also the explanation of the *offset* parameter for relative and absolute offsets.

0 Relative offset specified in the *offset* parameter (default)
1 Absolute offset specified in the *offset* parameter

Default: 0

offset

32-bit signed integer by value (optional)

Passes an integer representing a page offset, either absolute or relative, within the spool file. Together with the *direction* parameter, it tells the spooler where to resume when the file is picked up again for printing.

If "absolute" is specified in *direction*, printing resumes at that page, absolute from the beginning of the file. If "relative" is specified in *direction*, then depending on whether *offset* is positive or negative, the offset is adjusted either forward or backward relative to the current location, by the number of pages specified.

No matter which combination of offsets is specified, the final location is limited by the bounds of the file. The default for offset is 0. If the printing of a spool file is to be resumed from the beginning of the file, pass absolute for *direction* and 0 for *offset*.

Default: 0

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

Passes a value that indicates whether the spooling queue is to be opened or disabled when the spooling process is suspended. The default is not to change the current *q_state* of the spooler process. The valid inputs and their meanings are as follows:

0	No change to the current *q_state* of the spooling process (default)
1	Openq
2	Shutq

Default: 0

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

Operation Notes

`AIFSPPSUSPEND` should be called only when the spooler is in the active or idle state, or to accelerate a previous `SUSPEND;FINISH` to a `SUSPEND;NOW`. `AIFSPPSUSPEND` is the programmatic interface for executing the commands `SUSPENDSPOOL` and `SPOOLER` *dev*`;SUSPEND`.

AIFSYSWIDEGET

Returns system information (for example, PIDs and UFIDs) to be used as input keys by other AIFs to access more detailed information.

Syntax

```
                        REC          I32          A            A
AIFSYSWIDEGET (overall_status, aif_area, return_array1, return_array2,
                        I32          I32A         @64A
               num_entries, itemnum_array, item_array,
                        RECA         REC      I32
               itemstatus_array, search_key, user_id,
                        @64
               buffer_ptr);
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in ***itemstatus_array***, signaling an error condition. Refer to appendix A for meanings of status values.

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

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

Passes a value indicating the information area (for example, process information or file information) for which system wide information is desired. Values indicating the desired information area are specified in Table 3-37.

return_array1 **array by reference (optional)**

Returns system information keys (for example, PIDs and UFIDs). The keys can also be used by other AIFs to access more detailed information associated with the key. The size and type of key is dependent on the information area specified in *aif_area*. The size and type of keys are specified in Table 3-37. If a nil address (the default value) is passed, no keys are returned.

Make the array large enough to hold the largest number of keys that you expect to receive.

Array type: (Refer to Table 3-37.)

Default: nil

return_array2 **array by reference (optional)**

Returns system information keys (for example, PIDs, UFIDs). The keys can also be used by other AIFs to access more detailed information associated with the key. The size and type of a key is dependent on the information area specified in *aif_area*. The size and type of keys are specified in Table 3-37. If a nil address (the default value) is passed, no keys are returned.

Make the array large enough to hold the largest number of keys that you expect to receive.

Array type: (Refer to Table 3-37.)

Default: nil

num_entries **32-bit signed integer by reference (required)**
On input, the number of entries is the number of elements in *return_array1* or *return_array2*. On output, *num_entries* represents the number of elements returned in *return_array1*, *return_array2*, or the buffer pointed to by *buffer_ptr*. If *return_array1*, *return_array2*, and *buffer_ptr* were nil pointers, the returned value would be the number of instances meeting the specified item criteria.

Note that *num_entries* is only used on input when *return_array1* or *return_array2* are specified, but it is used on output when either *return_array1*, *return_array2*, or *buffer_ptr* are specified.

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

An array of integers where each element is an item number indicating the class of selection criteria located in the corresponding element in *item_array*. Valid items depend upon the information area specified in *aif_area*. For example, if information is desired about processes, then the criteria may be process-state, capabilities, and so on. If n criteria are specified, element $n+1$ must be a zero to indicate the end of the element list.

Refer to the `AIFSYSWIDEGET` Criteria Item Description tables for descriptions of item numbers and their corresponding items.

Default: nil

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

An array where each element is a 64-bit address pointing to a data structure containing the specific value of the criteria item specified in *itemnum_array*.

Refer to the `AIFSYSWIDEGET` Criteria Item Description tables for descriptions of item numbers and their corresponding items.

Array type: `globalanyptr`

Default: nil

itemstatus_array **record array by reference (optional)**

An array where each element returns the status about each of the selection criteria located in the corresponding element in *item_array*. For example, an error condition is returned if a criteria value in the corresponding element in *item_array* is incorrect, or if the criteria is no longer supported. (Refer to appendix A for status descriptions.)

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

Default: nil

search_key **record by reference (optional)**

In the event that *return_array1* and *return_array2* are not large enough to contain all the returned values of the specified criteria, a search key is returned. On a subsequent call to `AIFSYSWIDEGET`, the search key eliminates duplicating values that have already been returned. No search key is returned for spool files.

The initialization value of *search_key* is determined by the information area specified in ***aif_area*** prior to the first call to `AIFSYSWIDEGET`. The appropriate initialization values are specified in Table 3-37.

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

Default: nil

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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

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

A long pointer to a buffer which is sufficient to hold the items requested. The size of this buffer (in bytes) must be specified in the first four bytes of the buffer. On return from `AIFSYSWIDEGET`, these four bytes will contain the actual number of bytes returned.

See the `buffer_type` declaration in appendix B for a suggestion on one way to define this buffer when retrieving a list of HFS pathnames.

Operation Notes

The following information is specified in Table 3-37:

- A value corresponding to each specified information area to be passed in the ***aif_area*** parameter.
- The data type required to be passed in *return_array1* and/or *return_array2* that corresponds to the information area specified in the ***aif_area*** parameter.
- The data type and initialization value to be passed in the *search_key* parameter. The *search_key* for ascii strings must be initialized to blanks.

Table 3-37. AIFSYSWIDEGET Parameter Information

Area Value	Area Name	*return_array1* Type	*return_array2* Type	*search_key* Type Initial Value
1000	Job/session information	Job/session key (`jskey_type`)	Job/session number (`jsnum_type`)	(`I32`) 0
2000	Process information	PID (`longint_type`)	None	(`I32`) 0
5000	File information	MPE Files (item 5001) UFIDs (`UFID_type`) MPE and HFS Files (item 5036) Path identifier (path_identifier)	MPE Files (item 5001) File name (`filename_type`) MPE and HFS Files (item 5036) Buffer Information (`buffer_info_type`)	MPE Files (item 5001) Filename (`filename_type`) Blanks MPE and HFS Files (item 5036) Pathname (`max_pathname_type`) Blanks
6000	Accounting information	None	Directory name (`directory_name_type`)	`directory_name_type` Blanks
8000	Spool file information	Spool file address (`@64`)	Spool file number (`spf_id_type`)	Not applicable
13000	Device	LDEV Number (`I32`)	Device Key (`ufid_type`)	(`I32`) 0
13500	Device Class	Device Class Name (`C16`)	Device Class Key (`I32`)	(`I32`) 0
14000	Console Reply information	Reply request ID (`I32`)	None	(`I32`) 0
19000	Workgroup information	Workgroup Names (`CA256`)	None	key_wg_type (0)

Refer to appendix B for descriptions of the indicated data types.

If a criteria item is of type integer, a range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array*, and passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value the upper limit (<=).

Criteria items that have range capability are noted in the tables of `AIFSYSWIDEGET` criteria item descriptions.

Device Class Area The "Device Key" (item 13500) is the UFID of the selected device file. The "Device Class Key" (item 13000) is the class index to the Device Class Table. This is a faster key to access device class information than the Device Class Name.

File Area If you specify both item 5001 (MPE file name) and item 5036 (HFS pathname), item 5036 will be used when selecting files.

Item 5036, Return array 1 (Path Identifier)

If you are interested only in MPE files, the UFID key is the faster key to access file information using the `AIFFILEGET/PUT` AIFs. If you are interested in all files including both MPE syntax and HFS syntax files, the path identifier is the faster key to access file information. The path identifier contains the file UFID, link ID, and parent UFID. These items provide the necessary information to the HFS directory services to provide fast access.

Note that throughout the file related AIFs, UFID keys and UFID items are still valid for HFS syntax as well as MPE syntax files since a UFID is still unique for every file on the system. However, the UFID alone is not enough information to identify a unique filename for an HFS syntax file since the filename is no longer kept in the file label and since multiple file links will be supported in the future.

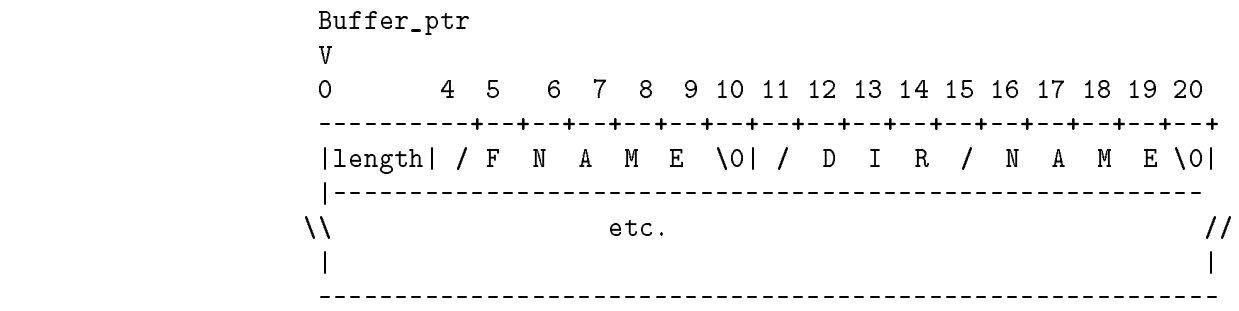
Item 5036, Return array 2 (Buffer Information)

Return array 2 is defined as an array of entries where each entry contains the following:

- Buffer offset - Buffer_ptr relative offset to pathname.
- Pathname length - Length of the pathname. The length does not include the NULL terminator.

The information in this return array can be used to index directly into the buffer of pathnames if you wish to perform binary searches for example. It can also be used in conjunction with a Pascal STRMOVE to easily retrieve a pathname from the buffer.

The following diagram illustrates this:



Return Array 2

Note

Although chaining (that is, calling `AIFSYSWIDEGET` multiple times passing it a search key) is supported for item 5036, it is not recommended for best performance results. It will be much faster to retrieve as many files as possible with fewer calls to `AIFSYSWIDEGET`.

Operation Notes - HFS Pathname Syntax

`AIFSYSWIDEGET` will accept a generic pathname using the same wildcarding as the `LISTFILE` command. See the description of the `LISTFILE` command in the *MPE/iX Reference Supplement* (32650-90353) for more information.

Below are some of the rules defining the syntax of an HFS pathname.

- Directory names end in a slash (/). This includes MPE accounts, MPE groups, and HFS directories.
- A filename can have a maximum of 255 characters.
- If the fileset begins with a slash (/), then the pathname is assumed to be an absolute pathname.

- If the fileset begins with a ./, then the pathname is assumed to be a relative pathname.

Operation Notes - HFS Directory Security

With the new `AIFSYSWIDEGET` HFS file item, you must have the appropriate security access rights to traverse (TD - traverse directory) and read (RD - read directory) directories protected by an ACD. This is because the lower level directory traversal routines enforce security checking. This will enable you to provide your own `LISTF` type of application and take advantage of the system security checking without having to implement your own security routines.

System managers are always granted all access to files and hierarchical directories protected by ACDs. If your process does not already have System Manager (SM) capability, you can easily retrieve SM capability for the process by calling `AIFPROCPUT` with item 2095 to alter user capabilities.

See the description of the `ALTSEC` command in the *MPE/iX Reference Supplement* (32650-90353) and *Controlling System Activity* (32650-90155) for more information on file ACDs and system security.

Operation Notes - Handling Directory Traversal Errors

Starting in release 5.0, users have more flexibility in handling errors detected during directory traversal (valid only with HFS traversal). A new item, 5050, has been added to allow users to ignore non-fatal directory errors which will not prevent them from continuing traversal. When a non-fatal error is detected and this item is true, the directory traversal will continue without reporting any error to the user, and the file for which an error was detected will not be returned to the user's buffer.

If the user does not wish to ignore the error, then an error will be returned to the user, and the file for which the error was detected will be returned in the SEARCH KEY parameter. The user's buffer and return arrays would also contain all the files up until the point where the error was detected. The user can then handle the error if they wish, and then call `AIFSYSWIDEGET` again with the SEARCH KEY to continue the directory traversal starting with the NEXT file.

Some examples of non-fatal errors are bad UFIDs and lack of the appropriate security access rights (for example, no TD access).

Note

Some errors (for example, bad UFID) will now only be detected if you pass in criteria such as record type or file type which requires AIFSYSWIDEGET to retrieve the file label pointer and look in the file label for a matching criteria. This is like doing a LISTFILE as compared to a LISTFILE,-3 (for example, LISTFILE,-3 will report an error when there is a bad UFID, but LISTFILE will not). This change will improve performance since there is no need to go to the file label in all cases.

Programming Examples

Following are programming examples for AIFSYSWIDEGET.

Example One - Job Numbers

Following is an example of a call to AIFSYSWIDEGET to obtain a list of job numbers of all jobs on the system that are suspended.

```
Procedure AIFSYSWIDEGET(
  overall_status,
  aif_area,          = [ 1000 {Job/Sessions} ]
  ,
  return_array2,     = [ array of integer [1..n] ]
  num_entries,       = [ number of elements n ]
  itemnum_array,     = [ [1]=1002 {JOBSTATE},[2]=0 {TERMINATOR}]
  item_array,        = [ [1] = 4 {SUSPENDED} ]
  itemstatus_array,
  ,                    { no search key }
  user_id      );
```

Example Two - PIDs of CI processes

Following is an example of a call to AIFSYSWIDEGET to obtain a list of PIDs of all CI processes. A search key is returned if there are more PIDs than the number of elements in the *return_array1*.

```
Procedure AIFSYSWIDEGET(
  overall_status,
  aif_area,          = [ 2000 {PROCESS} ]
  return_array1,     = [ array of longint_type [1..n] ]
  ,
  num_entries,       = [ number of elements n ]
  itemnum_array,     = [ [1]=2151 {Process Type},[2]=0 {TERMINATOR} ]
  item_array,        = [ [1] = 2 {MAIN} ]
  itemstatus_array,
  search_key,        = [ initialize before first call ]
  user_id      );
```

Example Three - Number of output spool files

Following is an example of a call to AIFSYSWIDEGET to obtain the number of output spool files with priority greater than 7.

```
Procedure AIFSYSWIDEGET(
  overall_status,
  aif_area,          = [ 8000 {SPOOL FILES} ]
  ,
  ,
  num_entries,       = [ 0 ]
  itemnum_array,     = [ [1] = 8502 {Output Priority}, [2]=8502, [3]=0 ]
  item_array,        = [ [1] = 8, [2] = 14 ]
  itemstatus_array,
  ,                  { no search key }
  user_id            );
```

Example Four - List of accounts with SM capability

Following is an example of a call to AIFSYSWIDEGET to obtain a list of all accounts with SM capability:

```
Procedure  AIFSYSWIDEGET(
   overall_status,
   aif_area,          = [ 6000 {ACCOUNTING} ]
   ,
   return_array2,     = [ array of directory name types [1..n] ]
   num_entries,       = [ number of elements n ]
   itemnum_array,     = [ [1]=6203 {Account Capabilities}, [2] =0]
   item_array,        = [ [1] = an integer with bit16 enabled ]
   itemstatus_array,
   ,                  { no search key }
   user_id          );
```

Example Five - Configured devices for a device class

Following is an example of a call to AIFSYSWIDEGET to obtain a list of configured devices for a device class.

```
Procedure  AIFSYSWIDEGET(
   overall_status,
   aif_area,          = [ 13000 {DEVICE} ]
   return_array1,     = [ array of integer {list of LDEV numbers} ]
   ,
   num_array_entries, = [ number of elements in return_array ]
   itemnum_array,     = [ [1] = 13002 {device class} ]
   item_array,        = [ [1] = TERM ]
   item_status_array,
   search_key,
   user_id     );
```

Example Six - Configured devices for a range of LDEV numbers

Following is an example of a call to AIFSYSWIDEGET to obtain a list of configured devices for a range of LDEV numbers.

```
Procedure  AIFSYSWIDEGET(
           overall_status,
           aif_area,          = [ 13000 {DEVICE} ]
           return_array1,     = [ array of integer {list of LDEV numbers} ]
           ,
           num_array_entries, = [ number of elements in return_array ]
           itemnum_array,     = [ [1] = 13002, [2] = 13001  {a range of
                                                            LDEV numbers} ]
           item_array,        = [ [1] = 1, [2] = 20 {from LDEV #1 to #20} ]
           item_status_array,
           ,                    {no search key}
           user_id     );
```

Example Seven - Configured devices on the system

Following is an example of a call to AIFSYSWIDEGET to obtain a list of configured devices on the system.

```
Procedure  AIFSYSWIDEGET(
           overall_status,
           aif_area,          = [ 13000 {DEVICE} ]
           return_array1,     = [ character array {list of device classes} ]
           ,                    { do not want list of device keys }
           num_array_entries, = [ number of elements in return_array ]
           ,                    { no itennum }
           ,                    { no item_array }
           ,                    { no item_status_array }
           search_key,          { initialize to zero or blanks before the 1st
                                  call}
           user_id     );
```

Example Eight - Device classes for a LDEV number

Following is an example of a call to AIFSYSWIDEGET to obtain a list of device classes for a LDEV number.

```
  Procedure  AIFSYSWIDEGET(
             overall_status,
             aif_area,          = [ 13000 {DEVICE} ]
             return_array1,     = [ array of integer {list of LDEV numbers} ]
             return_array2,     = [ array of integer {the corresponding fast
                                                      device keys}]
             num_array_entries, = [ number of elements in return_array ]
             itemnum_array,     = [ [1] = 13501, [2] = 0 {LDEV} ]
             item_array,        = [ [1] = 6 {LDEV 6} ]
             item_status_array,
             ,                    {no search key}
             user_id     );
```

Note For more programming examples of AIFSYSWIDEGET refer to Appendix C.

AIFSYSWIDEGET Item Descriptions

The following tables provide detailed descriptions of the item numbers associated with system wide information.

Job/Session Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with job/session criteria used by AIFSYSWIDEGET.

Table 3-38. AIFSYSWIDEGET Job or Session Criteria Item Descriptions

Item Number	Item Name (Data Type) Range Capability; Release First Available Description
1001	**Job name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions whose job names equal the specified criteria value. The 16-byte character array must contain the job name (left-justified and padded with blanks). A 16-character identifier given to a job or session. It is left-justified, all capitals, and padded with blanks. All blanks represent a job or session that does not have a job name.
1002	**Job state (I32) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions with a current state equal to the specified criteria value. Values and their meanings are as follows: 1 Introduced (INTRO) 2 Executing (EXEC) 3 Terminating (TERM) 4 Suspended (SUSP) 32 Waiting (WAIT) 40 Error (ERROR) 48 Initializing (EXEC*) 56 Scheduled (SCHED) If a job or session is in the INIT state, there is no guarantee that any of the values returned by AIFSYSWIDEGET are valid.
1007	**Input priority (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions whose input priorities (INPRI) equal the specified criteria value(s). A value must be in the range 0..15. (A value of 15 is equivalent to using the ;HIPRI option of the JOB command.) When a job's input priority is higher than the system jobfence, the system allows the job to execute. A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).

Table 3-38. AIFSYSWIDEGET Job or Session Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range Capability; Release First Available Description
1008	**Output priority (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions whose output priorities (`OUTPRI`) equal the specified criteria value(s). A value must be in the range 0..14. When a job's output priority is higher than the outfence of the output device, the spool file that is associated with the $STDLIST for that job is sent to the device. A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
1009	**User name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions logged on to the specified user. The format is a 16-byte character array containing the user name (left-justified and padded with blanks).
1010	**Group name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions logged on to the specified group. The format is a 16-byte character array containing the group name (left-justified and padded with blanks). Since the same group name can be used in multiple accounts, criteria item 1011 must be specified in the following element of the *itemnum_array*/*item_array* pair.
1011	**Account name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions logged on to the specified account. The format is a 16-byte character array containing the account name (left-justified and padded with blanks).
1016	**Executing priority (I32) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions logged in the specified queue. The executing priority translates to the base of the queue that the job or session is logged on to. Values and meanings are as follows: 100 BS queue 150 CS queue 200 DS queue 250 ES queue
1037	**Job/session number (REC) Range capability: Yes; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of jobs/sessions whose job/session numbers equal the specified criteria value(s). A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=). Record type: `jsnum_type` (Refer to appendix B.)

Table 3-38. AIFSYSWIDEGET Job or Session Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range Capability; Release First Available Description
1039	**Session? (B) Range capability: No; Release 3.0** Passing this criteria returns the job/session keys and/or job/session numbers of either jobs or sessions. Values and their meanings are as follows: False Jobs True Sessions
1043	**HP DTC Portid (CA17) Put:No; Verify:Yes; Release 4.0** Returns the hpdtcportid system variable in SHOWVAR format. The format of hpdtcportid is DTC LAN station address followed by SIC and port numbers: 0800090001111 0002.
1044	**Job submitter job/session number (REC) Put:No; Verify:Yes; Release 4.0** Passing this criteria returns either of both of the keys and numbers for the job or session matching the submitter job or session number.
1045	**Job submitter job/session name (CA16) Put:No; Verify:Yes; Release 4.0** Passing this criteria returns either of both of the keys and numbers for the job or session matching the submitter job or session name.
1046	**Job submitter user name (CA16) Put:No; Verify:Yes; Release 4.0** Passing this criteria returns either of both of the keys and numbers for the job or session matching the submitter job or session user name.
1047	**Job submitter account name (CA16) Put:No; Verify:Yes; Release 4.0** Passing this criteria returns either of both of the keys and numbers for the job or session matching the submitter job or session account name.

Process Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with process criteria used by `AIFSYSWIDEGET`.

Table 3-39. AIFSYSWIDEGET Process Criteria Item Descriptions

Item Number	Item Name (Data Type) Range capability; Release First Available Description
2015	**Job/session number (REC) Range capability: No; Release 3.0** Passing this criteria returns the PIDs of processes related to the specified job/session number. Record type: `jsnum_type` (Refer to appendix B.)
2016	**Scheduling state (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the PIDs of processes with a process state (as viewed by the dispatcher) equal to the specified value(s). It is the first item that should be interrogated to ascertain a process's state. Valid values and their meanings are as follows: 0 Executing (only for Calling Process) 1 Ready 2 Short wait 3 Long wait 4 Null The processes in short wait and ready are linked together in the order of priority. A short wait is basically a wait for disk I/O, and the dispatcher expects the process to be ready in a short while. A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).

Table 3-39. AIFSYSWIDEGET Process Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
2017	**Scheduling queue (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the PIDs of processes belonging to the specified scheduling queue(s). Valid values and their meaning are as follows: 0 AS queue 1 BS queue 2 CS queue 3 DS queue 4 ES queue A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
2019	**Priority (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the PIDs of processes with priority equal to the specified criteria value(s). MPE/iX priorities are values in the range 0..32767. An MPE/iX priority is inverted with respect to an MPE V/E priority in that high MPE/iX priority values indicate higher priority. The conversion to MPE V/E priority can be accomplished as follows: `MPEVpri = (32767 - MPEiXpri) div 128 (all values are decimal)` MPE/iX priorities are very transient for user processes. For processes whose priority is not fixed, this value is interpreted as the priority at which the process was last dispatched. For nonconstant priority processes, this value has no bearing on the priority at which the process is next dispatched. A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
2033	**Process type (I32) Range capability: No; Release 3.0** Passing this criteria returns the PIDs of processes with process type equal to the specified criteria value. Values and their meanings are as follows: 0 User (any process created by a user) 1 Son (process created by CI to run user programs) 2 Main (CI process) 3 Task (not in use) 4 System (some integral processes) 5 Detach (not connected to the PROGEN tree) 6 UCOP (JSmain) 7 Unknown (uninitialized processes)

Table 3-39. AIFSYSWIDEGET Process Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
2065	**Open file (REC) Range capability: No; Release 3.0** Passing this criteria returns the PIDs of processes accessing the specified file. Valid only for NM files. Record type: `ufid_type` (Refer to appendix B.)
2070	**Capabilities (I32) Range capability: No; Release 3.0** Passing this criteria returns the PIDs of all processes that have the specified capabilities. For example, if bit (25:1) is set to 1, PIDs of all processes that have PM capability are returned. Bits (0:23) Unused Bit (23:1) Batch access Bit (24:1) Interactive access Bit (25:1) Privileged mode Bit (26:2) Unused Bit (28:1) Multiple RINs Bit (29:1) Unused Bit (30:1) Extra data segment Bit (31:1) Process handling
2144	**Workgroup name (CA256) Range capability: No** Passing this criteria returns all the PIDs of the processes who are natural or artificial members of the specified workgroup name. The workgroup name should be terminated by a NULL character (ASCII 0).

File Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with file criteria used by AIFSYSWIDEGET.

Table 3-40. AIFSYSWIDEGET File Criteria Item Descriptions

Item Number	Item Name (Data Type) Range capability; Release First Available Description
5001	**MPE file name (REC) Range capability: No; Release 3.0** Passing this criteria returns the UFID and/or file name of all files whose file names equal the specified criteria value. The name in each element of the record `filename_type` must be left-justified and padded with blanks. In addition, characters must be in the correct case (upper and/or lower). Use of @'s for wild carding is permitted. @ will default to home group and account. @.@ will default to home account. @.@.@ will get all files. Use item 5036 if you are interested in both MPE and HFS files. This item can only be used for files that can be represented by MPE-semantics. Record type: `filename_type` (Refer to appendix B.)
5008	**File code (I32) Range capability: No; Release 3.0** Passing this criteria returns the UFID and/or file name of all files whose file codes match the specified criteria value.
5013	**Privileged level (I32) Range capability: No; Release 3.0** Passing this criteria returns the UFID and/or file name of all files whose privileged level equals the specified criteria value. The valid range is 0..3 where 0 is the highest privileged level and 3 is the lowest.
5036	**HFS pathname (REC) Range capability: No; Release 4.5** Passing this criteria returns the path identifier and/or pathname of all files whose pathname meets the specified criteria value. The pathname in the record `pathname_type` must be left justified and padded with blanks. On input, the length in the record specifies the array size in bytes. Pathnames will be returned into the buffer pointed to by the buffer_ptr parameter. They will be returned as absolute or relative pathnames depending on the syntax of the name you specify for this item. For example, if you specify the pathname './@' on input, the names returned will be relative to the CWD (for example, ./file). If you specify the pathname '/SYS/PUB/@' on input, then the names returned will be absolute pathnames (for example, /SYS/PUB/file). Record type: `pathname_type` (Refer to appendix B.)

Table 3-40. AIFSYSWIDEGET File Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
5039	**File type (U32) Range capability: No; Release 4.5** Passing this criteria returns the files (MPE filename or HFS pathname) and unique file identifiers (UFID or path_identifier) of all files whose file type meets the specified criteria. The following are the file types: 0 - ordinary 1 - ksam 2 - relative_io 3 - nm_ksam 4 - circular 5 - spool 6 - message 7 - resv 8 - cmfile 9 - dir_obj 10 - label_table 11 - xm_syslog 12 - pipe 13 - fifo 14 - symbolic link 15 - device link
5040	**Record type (U32) Range capability: No; Release 4.5** Passing this criteria returns the files (MPE filename or HFS pathname) and unique file identifiers (UFID or path_identifier) of all files whose record type meets the specified criteria. The following are the record types: 0 - fixed 1 - variable 2 - undefined 3 - cm_spool 4 - account directory node 5 - user directory node 6 - group directory node 7 - fileset directory node 8 - temporary directory 9 - byte stream 10 - hierarchical directory
5049	**Recursion level (I32) Range capability: No; Release 4.5** Passing this criteria specifies the number of directory levels you wish to traverse in the hierarchical directory. Specify the value 0 to list only those files in the current level. The default is infinite traversal; that is, traverse all directories and sub-directories. This item is ignored if using item 5001.

Table 3-40. AIFSYSWIDEGET File Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
5050	**Ignore non-fatal errors? (B) Range capability: No; Release 5.0** Specifies whether or not the directory traversal should continue even if an non-fatal error is detected. An example of a non-fatal error is if a bad UFID is detected of if the user does not have the appropriate security (no TD) to traverse a directory. If this item is FALSE and a non-fatal error is detected, the directory traversal will stop, the error will be returned in the item status array, and the file where the error was detected will be returned in the search key. The user can then process the error and continue by calling `AIFSYSWIDEGET` again with the search key. The default is FALSE.

Accounting Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with accounting criteria used by AIFSYSWIDEGET.

Table 3-41. AIFSYSWIDEGET Accounting Criteria Item Descriptions

Item Number	Item Name (Data Type) Range capability; Release First Available Description
6001	**User name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the directory names whose user names equal the specified criteria value. Since the same user name may be used in multiple accounts, criteria item 6201 must be specified. The format is a 16-byte character array containing the identifier of the user name (left-justified and padded with blanks). Use of @'s for wild carding is permitted.
6003	**Capabilities (I32) Range capability: No; Release 3.0** Passing this criteria returns the directory names that have the specified user capabilities. Item 6001 must also be specified. For example, if bit (0:1) is set to 1, all directory names (with user names indicated by criteria 6001) that have SM capability are returned. Bit (0:1) SM Bit (1:1) AM Bit (2:1) AL Bit (3:1) GL Bit (4:1) DI Bit (5:1) OP Bit (6:1) CV Bit (7:1) UV Bit (8:1) LG Bit (9:1) SP Bit (10:1) PS Bit (11:1) NA Bit (12:1) NM Bit (13:1) CS Bit (14:1) ND Bit (15:1) SF Bits (16:7) Unused (set to 0) Bit (23:1) BA Bit (24:1) IA Bit (25:1) PM Bits (26:2) Unused (set to 0) Bit (28:1) MR Bit (29:1) Unused (set to 0) Bit (30:1) DS Bit (31:1) PH

Table 3-41. AIFSYSWIDEGET Accounting Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
6008	**Local attributes (I32) Range capability: No; Release 3.0** Passing this criteria returns the directory names whose group user-definable attributes equal the specified criteria value. Criteria items 6001 and 6201 must also be specified.
6101	**Group name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the directory names whose group names equal the specified criteria value. Since the same group name may be used in multiple accounts, criteria item 6201 must be specified. The format is a 16-byte character array containing the identifier of the group name (left-justified and padded with blanks). Use of @'s for wild carding is permitted.
6103	**Group capabilities (I32) Range capability: No; Release 3.0** Passing this criteria returns the directory names that have the specified group capabilities. Item 6101 must also be specified. For example, if bit (31:1) is set to 1, all directory names (with group names indicated by criteria 6101) that have PH resource capability are returned. Bits (0:23) Unused (set to 0) Bit (23:1) BA Bit (24:1) IA Bit (25:1) PM Bits (26:2) Unused (set to 0) Bit (28:1) MR Bit (29:1) Unused (set to 0) Bit (30:1) DS Bit (31:1) PH

Table 3-41. AIFSYSWIDEGET Accounting Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
6201	**Account name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the directory names whose account names equal the specified criteria value. The format is a 16-byte character array containing the identifier of the account name (left-justified and padded with blanks). Use of @'s for wild carding is permitted.
6203	**Account capabilities (I32) Range capability: No; Release 3.0** Passing this criteria returns the directory names that have the specified account capabilities. Item 6201 must also be specified. For example, if bit (0:1) is set to 1, all directory names (with account names indicated by criteria 6201) that have SM capability are returned. Bit (0:1) SM Bit (1:1) AM Bit (2:1) AL Bit (3:1) GL Bit (4:1) DI Bit (5:1) OP Bit (6:1) CV Bit (7:1) UV Bit (8:1) LG Bit (9:1) SP Bit (10:1) PS Bit (11:1) NA Bit (12:1) NM Bit (13:1) CS Bit (14:1) ND Bit (15:1) SF Bits (16:7) Unused (set to 0) Bit (23:1) BA Bit (24:1) IA Bit (25:1) PM Bits (26:2) Unused (set to 0) Bit (28:1) MR Bit (29:1) Unused (set to 0) Bit (30:1) DS Bit (31:1) PH
6214	**Local attributes (I32) Range capability: No; Release 3.0** Passing this criteria returns the directory names whose account user-definable attributes equal the specified criteria value. Criteria item 6201 must also be specified.

Spool File Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with spool file criteria used by AIFSYSWIDEGET.

Table 3-42. AIFSYSWIDEGET Spool File Criteria Item Descriptions

Item Number	Item Name (Data Type) Range capability; Release First Available Description
8501	**File state (I32) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose states equal the specified criteria value. Values and their meanings are as follows: 0 Open state (job/data input spool file being accessed) 1 Active state (job/data input spool file being created) 2 Create state (output spool file being created) 3 Defer state (defer option specified for output spool file) 4 Ready state (spool file ready to be input or output) 5 Transfer state (output spool file being transferred to remote node) 6 Print state (output spool file being printed on a device) 7 Problem state (abnormality preventing output spool file from printing) 8 Del_pending state (output spool file to be deleted after closing) 9 Spsave state (output spool file copies printed, SPSAVE option specified) 10 (Reserved)
8502	**Priority (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose output priorities equal that of the specified criteria value(s). A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
8504	**Disposition (I32) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of either of the following: ■ Spool files that are to be save after they are printed ■ Spool files that are to be purged after they are printed Values and their meanings are as follows: 1 Save after printing 2 Purge after printing

Table 3-42. AIFSYSWIDEGET Spool File Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
8509	**STDLIST of aborted job (I32) Range capability: No; Release 3.0** Passing this criteria returns spool file address and/or spool file number of either of the following: ■ Spool files that are the `$STDLIST` of an aborted job ■ Spool files that are not the `$STDLIST` of an aborted job Values and their meanings are as follows: 0 Not `$STDLIST` of an aborted job 1 `$STDLIST` of an aborted job
8511	**Copies (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose total number of copies to be printed equals the specified criteria value(s). A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
8512	**Ready date (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose created dates equal the specified criteria value(s). The format in the 32-bit integer is the same as that returned by the `CALENDAR` intrinsic. The format of the data passed is as follows: Bits (0:16) Unused (set to 0) Bits (16:7) The year of the century Bits (23:9) The day of the year A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
8514	**Number of pages (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose number of pages equal the specified criteria value(s). A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
8516	**User name and account name of creator (CA32) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose creator user and account names equal the specified criteria value. The first 16 bytes hold the user name, and the second 16 bytes hold the account name. The names should be left-justified and padded with blanks. Only the first 8 bytes of each field is used.

Table 3-42. AIFSYSWIDEGET Spool File Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
8517	**Job/session # (REC) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose creator job/session numbers equal the specified criteria value. The format of the data passed is as follows: Bits (0:2) Job or session (see below) Bits (2:30) The job/session number The values of bits (0:2) and their meanings are as follows: 0 Session not current to the system 1 Session current to the system 2 Job current to the system 3 Job not current to the system Record type: `Bit32` (Refer to appendix B.)
8518	**Job name (CA16) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose creator job names equal that of the specified criteria value.
8519	**File designator (CA16) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose formal file designators equal the specified criteria value.
8520	**Target device (REC) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files whose destination logical device number (LDEV), device name, and device class equal the specified criteria value. Record type: `device_name_type` (Refer to appendix B.)
8525	**Forms ID (CA16) Range capability: No; Release 3.0** Passing this criteria returns the output spool file address and/or spool file number of spool files whose forms IDs equal the specified criteria value.

Table 3-42. AIFSYSWIDEGET Spool File Criteria Item Descriptions (continued)

Item Number	Item Name (Data Type) Range capability; Release First Available Description
8528	**Number of records (I32) Range capability: Yes; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of spool files with the number of records equal to the specified criteria value(s). A range of values can be requested by passing the same criteria item number in consecutive elements of *itemnum_array* and by passing the lower and upper limits in the corresponding consecutive elements of *item_array*. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
8600	**Input/output (I32) Range capability: No; Release 3.0** Passing this criteria returns the spool file address and/or spool file number of either input spool files or output spool files. Values and their meanings are as follows: 1 Return input spool files 2 Return output spool files

Device Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with device criteria used by AIFSYSWIDEGET.

Table 3-43. AIFSYSWIDEGET Device Criteria Item Descriptions

Item Number	Item Name (Data Type) Range Capability; Release First Available Description
13001	**Logical Device Number (I32) Range Capability: Yes; Release 4.0** This is the LDEV number for the device. Specifying a LDEV number will return the LDEV number and the device key if it is configured. Specifying a range of LDEV numbers will return a list of configured LDEV's and the device keys within that range. A range of LDEV numbers can be requested by passing the same criteria item number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value will be the lower limit (>=) and the second value, the upper limit (<=).
13002	**User-Defined Device Class Name (C16) Range Capability: No; Release 4.0** This is the name of the device class assigned by the user using the I/O configurator command ACLASS in SYSGEN. The name is in upper-case, left-justified and padded with blanks to the right. Use of @'s for wild carding is permitted (for example, '@TAPE@ '). Specifying a device class will return the configured LDEV numbers and the device keys belonging to that device class.

Device Class Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with device class criteria used by AIFSYSWIDEGET.

Table 3-44. AIFSYSWIDEGET Device Criteria Item Descriptions

Item Number	Item Name (Data Type) Range Capability; Release First Available Description
13501	**Logical Device Number (I32) Range Capability: Yes; Release 4.0** This is the LDEV number for the device. Specifying a LDEV number will return the user-defined device classes of which the LDEV belongs and the corresponding device class keys. Specifying a range of LDEV numbers will return a list of user-defined device classes and device keys for each LDEV within that range. A range of LDEV numbers can be requested by passing the same criteria item number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value will be the lower limit (>=) and the second value, the upper limit (<=).
13502	**User-Defined Device Class Name (C16) Range Capability: No; Release 4.0** This is the name of the device class assigned by the user using the I/O configurator command ACLASS in SYSGEN. The name is in upper-case, left-justified and padded with blanks to the right. Use of @'s for wild carding is permitted (for example, '@TAPE@ '). Specifying a device class or a wildcarded device class will return the device class(es) and the device class key(s).

Console Reply Information Criteria Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with console reply criteria used by AIFSYSWIDEGET.

Table 3-45. AIFSYSWIDEGET Console Reply Information Criteria Item Descriptions

Item Number	Item Name (Data Type) Range Capability; Release First Available Description
14002	**Process Type (I32) Range capability:No; Release 4.0** Passing this criteria returns all reply request ids associated with either of the following: ■ System Processes ■ User Processes Values and their meanings are as follows: 0 System Process 1 User Process
14003	**Creation Time (I32) Range capability:Yes; Release 4.0** Passing this criteria returns all reply request ids for the creation time passed in. The format that can be passed is the same as returned by the CLOCK Intrinsic. Bits (0:8) The hour of the day Bits (8:8) The minute of the hour Bits (16:8) The seconds Bits (24:8) The tenths of seconds
14004	**Job/Session Number (REC) Release 4.0** Passing this criteria returns all reply request ids associated with the job or session number that is passed in. This is only valid for user processes. The format of Job or Session Number is: Bits (0:2) Job or session (1=session, 2=job) Bits (2:30) Job or session Number
14005	**Pin of the request (I32) Release 4.0** Passing this criteria returns all reply request ids associated with the pin that is passed in.

Workgroup Criteria Item Descriptions

The following table provides detailed descriptions of the item numbers and corresponding items associated with the new criteria, *workgroup* used by `AIFSYSWIDEGET`. The return value for the workgroup area is workgroup name(s).

Table 3-46. AIFSYSWIDEGET Workgroup Criteria Item Descriptions

Item Number	Item Name, Data Type, and Description
19001	**Workgroup name (CA256) Wildcarding capability: No** Passing this criteria returns the name of the workgroups that match the passed workgroup name. The name must be left-justified and terminated by a NULL character (ASCII 0). Use the @ symbol to represent all the workgroups.
19003	**Logon/User specification (CA256) Wildcarding capability: No** Passing this criteria returns all the workgroups that have the specified jobname, username.acctname as one of its membership criteria. The logon must be left-justified and terminated by a NULL character (ASCII 0). Use the @ symbol to represent all logons on the system. Only one logon/user can be specified. For example narinder,mgr.aiftest
19004	**Program/File name (REC) Wildcarding capability: No** Passing this criteria returns all the workgroups that have the specified file as one of its membership criteria. The name must be left-justified. The length of the item passed must be specified in the length field of the pathname_type record. Use the @ symbol to represent all files on the system. Only one program/file name can be specified. For example, Editor.pub.sys Record Type: `pathname_type`. The maximum size of n which is user-defined is 512.
19005	**Queue (CA20);** Passing this criteria returns all the workgroups that have the specified queue as one of its membership criteria. The queue must be left-justified and terminated by a NULL character (ASCII 0). Use the @ symbol to represent all queues of the set. Only one queue can be specified. For example, CS

Table 3-46. AIFSYSWIDEGET Workgroup Criteria Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19006	**Queue (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified queue as one of its membership criteria. Values and their meanings are as follows: 0 AS Queue 1 BS Queue 2 CS Queue 3 DS Queue 4 ES Queue A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=). The queue is represented as a character array in AIFWGADD and AIFWGGET/PUT. In order to have ranging capabilities, queue is represented by integers in AIFSYSWIDEGET. See item 19005 of AIFWGADD and AIFWGGET/PUT.
19007	**Base Priority (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified base priority value(s). MPE/iX priorities are values in the range 0..32767. An MPE/iX priority is inverted with respect to an MPE V/E priority in that in MPE/iX higher priority values indicate higher priority. The conversion to MPE V/E priority can be accomplished as follows: MPEVpri = (32767 - MPEiXpri) div 128 (all values are decimal) A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
19008	**Limit Priority (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified limit priority value(s). MPE/iX priorities are values in the range 0..32767. An MPE/iX priority is inverted with respect to an MPE V/E priority in that in MPE/iX higher priority values indicate higher priority. The conversion to MPE V/E priority can be accomplished as follows: MPEVpri = (32767 - MPEiXpri) div 128 (all values are decimal) A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).

Table 3-46. AIFSYSWIDEGET Workgroup Criteria Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19009	**Minimum Quantum (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified minimum quantum value(s). Values for minimum quantum range from 0 to 32767 milliseconds. A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
19010	**Maximum Quantum (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified maximum quantum value(s). Values for maximum quantum range from 0 to 32767 milliseconds. A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
19011	**Timeslice (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified timeslice value(s). Values for timeslice range from 100 to 32700 A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
19012	**Boost Property (I32) Range capability: No** Passing this criteria returns all the workgroups that have the specified boost property value. Values and their meanings are as follows: 0 Decay 1 Oscillate Since Boost Property has only two values, range capability is not needed. If boost property is not specified, it is ignored as search criteria.

Table 3-46. AIFSYSWIDEGET Workgroup Criteria Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19013	**Minimum CPU Percentage (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified Minimum CPU Percentage value(s). The value can range from 0% to 100%. A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).
19014	**Maximum CPU Percentage (I32) Range capability: Yes** Passing this criteria returns all the workgroups that have the specified Maximum CPU Percentage value(s). The value can range from 0% to 100%. A range of values can be requested by passing the same criteria number in consecutive elements of itemnum_array and by passing the lower and upper limits in the corresponding consecutive elements of item_array. The first value must be the lower limit (>=) and the second value, the upper limit (<=).

AIFTIME

Converts ticks or microseconds to a meaningful time such as date time, clock time, or a string format.

Syntax

```
                  REC          REC      REC     REC
AIFTIME (overall_status, ticks, microsecs, clock,
           REC     REC      I32          REC
         date, date_str, user_id, ticks_since_1970,
                  REC
         microsecs_since_1970);
```

Parameters

overall_status — **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: `status_type` (Refer to appendix B.)

ticks — **record by reference (optional)**

Passes a value, representing the ticks since 1970, that is to be converted to a meaningful time. If neither *ticks* nor *microsecs* is passed, the current time is assumed.

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

microsecs — **record by reference (optional)**

Passes a value, representing the microseconds since 1970, that is to be converted to a meaningful time. If neither *ticks* nor *microsecs* is passed, the current time is assumed.

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

clock — **record by reference (optional)**

Returns the time in hours, minutes, seconds, and tenths of seconds.

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

date **record by reference (optional)**

Returns the time in year, month, and day of month.

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

date_str **record by reference (optional)**

Returns the time in string format for month and day of the week. Record type: `datestr_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: Operating System product. If it is not passed, the caller must have previously called `AIFACCESSON`.

Default: 0

tics_since_1970 **record by reference (optional)**

Returns a value representing the current ticks since 1970.

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

microsecs_since_1970 **record by reference (optional)**

Returns a value representing the current microseconds since 1970.

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

Operation Notes

None.

AIFWGADD

Adds a new workgroup to the current list of workgroups.

Syntax

```
                          REC          I32A
AIFWGADD(overall_status, itemnum_array,
                        @64A          RECA              CA
                    item_array, itemstatus_array, workgroup_name,
                          CA          I32
                    position, user_id )
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the operating system information to be added. New information must be located in a data structure pointed to by the corresponding element in item_array. If n item numbers are being passed, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure containg new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

workgroup_name **256-byte character array by reference (required)**

Passes the name of the workgroup to be added. The workgroup name follows the convention for CI variables and Job Control Words(JCW's) and can be a maximum of 256 alphanumeric characters or underscores, where the first character cannot be numeric. The user-specified name (including case) is preserved. The workgroup_name is terminated by a NULL character (ASCII 0).

Note that the following names are unavailable: AS_Default, BS_Default, CS_Default, DS_Default, ES_Default, and Natural_wg.

position **256-byte character array by reference (optional)**

Passes the name of the workgroup where the new workgroup is to be inserted. The position is the name of an existing user-defined workgroup. The new workgroup will be inserted before the existing workgroup. The position specification is optional. If omitted, the new workgroup will be appended after all user-defined workgroups. Default workgroups cannot be specified in position parameter.

Default: Blanks

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

`AIFWGADD` adds a new workgroup in the current list of workgroups. When creating a new workgroup, it is necessary to specify one of the membership criteria and the required scheduling characteristics of Base and Limit. The rest of the membership criteria or scheduling characteristics can either be specified or allowed to take their default values. Table 4-6 provides detailed description of items. Tables 3-1 and 3-2 list the default values of membership criteria and scheduling characteristics.

Workgroup membership criteria is used to determine the workgroup membership of processes on the system. The determination is made at each process creation, whenever one of the values on which membership can be based is changed (Logon, Program, or Scheduling Queue), and whenever a system-wide scan is requested.

An addition of a new workgroup could effect the workgroup assignment of existing processes. As a result, the Workload Manager will need to scan all processes on the system and adjust their workgroup membership as necessary.

Workgroup Information Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with AIFWGADD

Table 3-47. AIFWGADD Item Descriptions

Item Number	Item Name, Data Type, and Description
19003	**Logon/User specification (CA256);** Passes the logon category of the new workgroup. The logon membership criteria specifies the job/session, user and account name (LOGON = *[job/sessionname,]username.accountname*) of potential workgroup members. The job/session name is optional, but if specified, the logon must be enclosed in double quotes (" "). The user and account name are required. Wildcarding is supported. Use the @ symbol to specify zero or more alphanumeric characters. For example: "SUSAN,MANAGER.SYS", GUEST.SYS, ACCNTING,@.SYS The logon/user specification must be left justified and terminated by a NULL character. "@,@.@" represents all logons on the system.
19004	**Program/File name (REC);** Passes the program file category of the specified workgroup. The program membership criteria specifies the program files of potential workgroup members. The filename must be a fully qualified MPE/iX file name or absolute Hierarchichal File System (HFS) name. Wildcarding is supported. Use the @ symbol to specify zero or more alphanumeric characters. Use the # symbol to specify one numeric character. For example: EDITOR.PUB.SYS, HPEDIT#.@.@ The name must be left-justified. The length of the item passed in must be specified in the length field of the pathname_type record. @.@.@ represents all files on the system. Record Type: pathname_type. The maximum size of n which is user-defined is 512.
19005	**Queue (CA20);** Passes the queue category of the specified workgroup. The queue membership criteria specifies the queue attribute of potential workgroup members. Five values are supported, AS, BS, CS, DS, and ES. For example: CS, ES The queue must be left justified and terminated by a NULL character. If a queue criteria is not specified, it defaults to match any of the five queues.

Table 3-47. AIFWGADD Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19007	**Base priority (I32);** Passes the base priority of the specified workgroup. This value is the highest priority that any process which is a member of this workgroup can have. Can be set for any user-defined workgroups. This priority can be mapped to MPE V by the following formula: MPEVPri = (32767 - MPE/iXPri) DIV 128 (All formula values are decimal.) Base priority is a required item for addition of a new workgroup.
19008	**Limit priority (I32);** Passes the limit priority of the specified workgroup. This value is the lowest priority that any process which is a member of this workgroup can have. Can be set for any user-defined workgroups. This priority can be mapped to MPE V by the following formula: MPEVPri = (32767 - MPE/iXPri) DIV 128 (All formula values are decimal.) Limit priority is a required item for addition of a new workgroup.
19009	**Minimum Quantum (I32);** Passes the minimum number of milliseconds of CPU consumption that is used to determine priority decay for processes within the specified workgroup. Can be set for any user-defined workgroups. Values for minimum quantum range from 0 to 32767 milliseconds. The default value is 1 milliseconds for user-defined workgroups and CS_Default workgroup. The default value for DS_Default and ES_Default workgroups is 2000.
19010	**Maximum Quantum (I32);** Passes the maximum number of milliseconds of CPU consumption that is used to determine priority decay for processes within the specified workgroup. Can be set for any user-defined workgroups. Values for maximum quantum range from 0 to 32767 milliseconds. The default value is 2000 milliseconds.
19011	**Timeslice (I32);** Passes the maximum amount of CPU time that can be consumed by a member of the specified workgroup before being timesliced (yielding the CPU). This value is accurate to 100-millisecond granularity and has a minimum value of 100 milliseconds. Values for timeslice range from 100 to 32700 and default value is 200 milliseconds for CS_Default, DS_Default, ES_Default and user-defined workgroups.
19012	**Boost Property (I32);** Passes the boost property of the workgroup (decay or oscillate). Values and their meanings are as follows: 0 Decay 1 Oscillate

Table 3-47. AIFWGADD Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19013	**Maximum CPU Percentage (I32);** Passes the upper bound for the amount of CPU the processes in a workgroup can consume relative to to other workgroups. The value can range from 0% to 100%. The default value is 100%. The maximum CPU percentage control may result in system idling if the workgroup hits its maximum CPU percentage and there are no other users who want CPU.
19014	**Minimum CPU Percentage (I32);** Passes the lower bound for the amount of CPU the processes in a workgroup can consume relative to to other workgroups. The value can range from 0% to 100%. The default value is 0%. Note that CPU consumption of the workgroup may not precisely match the specified the minimum CPU percentage if there is insufficient demand within the workgroup.

AIFWGGET

Returns workgroup information about a particular workgroup.

Syntax

```
                                REC          I32A
AIFWGGET(overall_status,  itemnum_array,
                               @64A          RECA
                          item_array,  itemstatus_array,
                                CA           I32
                          workgroup_name,  user_id)
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the information to be returned to a data structure pointed to in the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure where information is returned. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

workgroup_name **256-byte character array by reference (required)**

Passes the name of the workgroup for which information is desired. The workgroup name follows the convention for CI variables and Job Control Words(JCW's) and can be a maximum of 255 alphanumeric characters or underscores, where the first character cannot be numeric. The user-specified name (including case) is preserved, though comparisions are case insensitive. The workgroup_name is terminated by a NULL character (ASCII 0).

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

`AIFWGGET` requires workgroup name as an input parameter. It can be obtained by calling AIFSYSWIDEGET and specifying the work group area.

AIFWGPURGE

Purges a workgroup from the current list of workgroups.

Syntax

```
                                REC              CA
AIFWGPURGE( overall_status,  workgroup_name,
                                B            I32
                          purgescan,  user_id )
```

Parameters

overall_status **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.

Record type: status_type

workgroup_name **256-byte character array by reference (required)**

Passes the name of the workgroup to be deleted. The workgroup name follows the convention for CI variables and Job Control Words(JCW's) and can be a maximum of 255 alphanumeric characters or underscores, where the first character cannot be numeric. The user-specified name (including case) is preserved, though comparisions are case insensitive. All 255 characters are significant. The workgroup_name is terminated by a NULL character (ASCII 0).

Note that the following system workgroups cannot be purged:
AS_Default, BS_Default, CS_Default, DS_Default, and ES_Default.

purgescan **Boolean by reference (optional)**

Passes a boolean value denoting the deletion of the workgroup should cause a scan of **all processes of the deleted workgroup** in order to assign them to the new workgroups. The default does not cause any scanning of the processes.

This parameter when set to true will re-assign processes of **ALL PURGE-PENDING WORKGROUPS** to other workgroups.

Default: False

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

`AIFWGPURGE` deletes a workgroup from the current list of workgroups. A user can call AIFWGPURGE with the default "scan" option and the last call of AIFWGPURGE with purge-pending scan option. This will result in purge of all the requested workgroups and the last call to AIFWGPURGE will result in scanning of the processes of **ALL** the purged workgroups and their reassignment. This prevents scanning and reassignment of member processes at every workgroup purge.

When a workgroup is purged, the Workload Manager will rescan the affected member processes. The cost of such a rescan is dependent upon the number of processes, and workgroups, involved. If parameter "purgescan" is not passed the system does not complete the purge until all member processes have found a new workgroup. A workgroup in such a state is considered to have a purge pending. The scan of processes assigned to purge-pending workgroups is a subset of a system-wide scan. A system-wide scan checks all processes on the system for reassignment, while a purge-pending scan will only check processes that are assigned to purge-pending workgroups.

Logically, a workgroup in the purge-pending state no longer exists and thus can not accept new members. However, physically the workgroup remains until either its last member has died or been moved to another workgroup, or a scan is performed. When a workgroup goes into the purge-pending state, the system renames the workgroup by prepending the previous name with "~". The last character may be truncated to keep the new name to 255 characters.

AIFWGPUT

Modifies workgroup information.

Syntax

```
                        REC          I32A
AIFWGPUT(overall_status, itemnum_array,
                       @64A           RECA
                  item_array, itemstatus_array,
                        CA             I32
                  workgroup_name, user_id,
                       I32A           @64A          RECA
                  ver_item_nums, ver_items, ver_item_statuses )
```

Parameters

overall_status **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, not specific to any particular item. A positive value indicates the last element in itemstatus_array, signaling an error condition.

Record type: status_type

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

An array of integers where each element is an item number indicating the operating system information to be modified. New information must be located in a data structure pointed to by the corresponding element in item_array. If n item numbers are being requested, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure containg new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (required)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

workgroup_name **256-byte character array by reference (required)**

Passes the name of the workgroup whose information is to be modified. The workgroup name follows the convention for CI variables and Job Control Words(JCW's) and can be a maximum of 255 alphanumeric characters or underscores, where the first character cannot be numeric. The user-specified name (including case) is preserved, though comparisions are case insensitive. The workgroup_name is terminated by a NULL character (ASCII 0).

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

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

An array of integers where each element is an item number indicating the operating system information to be verified before proceeding with modification. Verification information must be located in a data structure pointed to by the corresponding element in ver_items. If n items are being verified, element n+1 must be a zero to indicate the end of the item list.

Default: nil

ver_items **64-bit address array by reference (optional)**

An array where each element is a 64-bit address pointing to a data structure containing information to be verified against current operating system information. Information and its required data type are defined by the item number passed in the corresponding element in ver_item_nums.

Array type: globalanyptr

Default: nil

ver_item_statuses **record array by reference (optional)**

An array where each element returns the status of the verification performed in the corresponding element in ver_items. A zero indicates a successful verification. A negative value indicates an error condition. A positive value indicates a warning.

Array type: status_type

Default: nil

Operation Notes

`AIFWGPUT` requires workgroup name as an input parameter. It can be obtained by calling AIFSYSWIDEGET and specifying the work group area. AIFWGPUT allows callers to modify scheduling characteristics of a particular workgroup.

Modification of scheduling characteristics does not cause scanning of processes as they do not effect workgroup membership.

Workgroup Information Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with Workload Manager workgroup information.

Table 3-48. Workgroup Information Item Descriptions

Item Number	Item Name, Data Type, and Description
19002	**Purge Pending? (B) Put: No; Verify: Yes** Returns a boolean value denoting whether a purge is pending for the indicated workgroup. When a workgroup is purged, the Workload Manager will need to rescan the affected member processes.
19003	**Logon/User specification (CA256) Put: No; Verify: Yes** Returns the logon category of the specified workgroup. The logon membership criteria specifies the job/session, user and account name (LOGON = *[job/sessionname,]username.accountname*) of potential workgroup members. The job/session name is optional, but if specified, the logon must be enclosed in double quotes (" "). The user and account name are required. Wildcarding is supported. The @ symbol specifies zero or more alphanumeric characters. For example: “SUSAN,MANAGER.SYS”, GUEST.SYS, ACCNTING,@.SYS The logon/user specification is left justified and terminated by a NULL character. “@,@.@” represents all logons on the system.
19004	**Program/File name (REC) Put: No; Verify: Yes** Returns the program file category of the specified workgroup. The program membership criteria specifies the program files of potential workgroup members. The filename must be a fully qualified MPE/iX file name or absolute Hierarchichal File System (HFS) name. Wildcarding is supported. The @ symbol specifies zero or more alphanumeric characters. The # symbol specifies one numeric character. For example: EDITOR.PUB.SYS, HPEDIT#.@.@ The name must be left-justified. The length of the item passed must be specified in the length field of the pathname_type record. @.@.@ represents all files on the system. Record Type: pathname_type. The maximum size of n which is user-defined is 512.
19005	**Queue (CA20) Put:No; Verify: Yes** Returns the queue category of the specified workgroup. The queue membership criteria specifies the queue attribute of potential workgroup members. Five values are supported, AS, BS, CS, DS, and ES. For example: DS, ES The queue is left justified and terminated by a NULL character.

Table 3-48. Workgroup Information Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19007	**Base priority (I32) Put: Yes; Verify: Yes** Returns or modifies the base priority of the specified workgroup. This value is the highest priority that any process which is a member of this workgroup can have. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; cannot modify the AS_Default or BS_Default workgroups. It can be set by the NEWWG or ALTWG commands for all workgroups except AS_Default and BS_Default workgroup. It can also be set by TUNE command for CS_Default, DS_Default or ES_Default workgroup. This priority can be mapped to MPE V by the following formula: MPEVPri = (32767 - MPE/iXPri) DIV 128 (All formula values are decimal.) Base priority is a required item for addition of a new workgroup.
19008	**Limit priority (I32) Put: Yes; Verify: Yes** Returns or modifies the limit priority of the specified workgroup. This value is the lowest priority that any process which is a member of this workgroup can have. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; cannot modify the AS_Default or BS_Default workgroups. It can be set by the NEWWG or ALTWG commands for all workgroups except AS_Default and BS_Default workgroup. It can also be set by TUNE command for CS_Default, DS_Default or ES_Default workgroup. This priority can be mapped to MPE V by the following formula: MPEVPri = (32767 - MPE/iXPri) DIV 128 (All formula values are decimal.) Limit priority is a required item for addition of a new workgroup.
19009	**Minimum Quantum (I32) Put: Yes; Verify: Yes** Returns or modifies the minimum number of milliseconds of CPU consumption that is used to determine priority decay for processes within the specified workgroup. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; does not exist for the AS_Default or BS_Default workgroups. It can also be set by the TUNE, NEWWG, or ALTWG commands. Values for minimum quantum range from 0 to 32767 milliseconds. The default value is 1 milliseconds for CS_Default workgroup and user-defined workgroups. The default value for DS_Default and ES_Default workgroups is 2000.
19010	**Maximum Quantum (I32) Put: Yes; Verify: Yes** Returns or modifies the maximum number of milliseconds of CPU consumption that is used to determine priority decay for processes within the specified workgroup. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; does not exist for the AS_Default or BS_Default workgroups. It can also be set by the TUNE, NEWWG, or ALTWG commands. Values for maximum quantum range from 0 to 32767 milliseconds. The default value is 2000 milliseconds for.

Table 3-48. Workgroup Information Item Descriptions (continued)

Item Number	Item Name, Data Type, and Description
19011	**Timeslice (I32) Put: Yes; Verify: Yes** Returns or modifies the maximum amount of CPU time that can be consumed by a member of the specified workgroup before being timesliced (yielding the CPU). This value is accurate to 100-millisecond granularity and has a minimum value of 100 milliseconds. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; cannot modify for the AS_Default or BS_Default workgroups. It can also be set by the TUNE, NEWWG, or ALTWG commands. Values for timeslice range from 100 to 32700 and default value is 200 milliseconds CS_Default, DS_Default, ES_Default and user-defined workgroups. The default value for AS_Default and BS_Default workgroups is 1000.
19012	**Boost Property (I32) Put: Yes; Verify: Yes** Returns or modifies the boost property of the workgroup (decay or oscillate). Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; does not exist for the AS_Default or BS_Default workgroups. It can also be set by the TUNE, NEWWG, or ALTWG commands. Values and their meanings are as follows: 0 Decay 1 Oscillate
19013	**Maximum CPU Percentage (I32) Put: Yes; Verify: Yes** Returns or modifies the upper bound for the amount of CPU the processes in a workgroup can consume relative to to other workgroups. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; does not exist for the AS_Default or BS_Default workgroups. It can also be set by the TUNE, NEWWG, or ALTWG commands. The value can range from 0% to 100%. The default value is 100%. The maximum CPU percentage control may result in system idling if the workgroup hits its maximum CPU percentage and there are no other users who want CPU.
19014	**Minimum CPU Percentage (I32) Put: Yes; Verify: Yes** Returns or modifies the lower bound for the amount of CPU the processes in a workgroup can consume relative to to other workgroups. Can modify for any user-defined workgroups, or the CS_Default, DS_Default, and ES_Default workgroups; does not exist for the AS_Default or BS_Default workgroups. It can also be set by the TUNE, NEWWG, or ALTWG commands. The value can range from 0% to 100%. The default value is 0%. Note that CPU consumption of the workgroup may not precisely match the specified the minimum CPU percentage if there is insufficient demand within the workgroup.
19015	**Quantum (I32) Put: No; Verify: No** Returns the average number of milliseconds that a process in the workgroup executes before it is interrupted. It is used to decide when a process in that workgroup should have its priority decayed. It is maintained dynamically by the system and is very transient in nature.

AIFWGREPLACE

Replaces the current set of workgroup(s) by a new set of workgroup(s) specified in the file.

Syntax

```
                                    REC          I32          I32A
AIFWGREPLACE(overall_status, file_num, itemnum_array,
                                  @64A          RECA          I32
                        item_array, itemstatus_array, user_id )
```

Parameters

overall_status **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.

Record type: status_type

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

Passes the file number of the ASCII file which defines the new set of workgroup(s).

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

An array of integers where each element is an item number indicating the operating system information to be added. New information must be located in a data structure pointed to by the corresponding element in item_array. If n item numbers are being passed, element n+1 must be a zero to indicate the end of element list.

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

An array where each element is a 64-bit address pointing to a data structure containg new information to be passed to the operating system. Information and its required data type are defined by the item number passed in the corresponding element in the itemnum_array.

Array type: globalanyptr

itemstatus_array **Record array by reference (optional)**

An array where each element returns the status of the operation performed in the corresponding element in item_array. A zero indicates a successful operation. A negative value indicates an error condition.

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: Operating System product. If it is not passed, the caller must have previously called AIFACCESSON.

Default: 0

Operation Notes

`AIFWGREPLACE` requires a file number as an input parameter. The file containing the workgroup specifications needs to be opened by the caller. The file should be an ASCII file (i.e., The file may be temporary or permanent and have fixed or variable length records). The file will contain specifications for creating user-defined workgroups. Workgroup creation will not begin until all specifications within the file have passed a syntax check. Furthermore, the system will consider the creation an atomic operation (i.e., either all workgroups within the file will be created or none). The file thus establishes a new set of workgroups, **deleting all existing workgroups.** This results in the creation of new workgroups, and the deletion of the old workgroups. The five default workgroups cannot be deleted; if they are not in the specified file, they will retain their existing characteristics.

If a semantic or syntax error occurs while processing the file, AIFWGREPLACE returns overall status less than zero. If the overall status error number is between the syntax or semantic error range, and the first three items (19501 to 19503) are specified, then the CI error information is returned in those items.

The specification for an **individual workgroup** is given below.

```
Workgroup =            workgrp
;MEMB_LOGON =          logon
;MEMB_PROFILE =        profile
;MEMB_PROGRAM =        program
;MEMB_QUEUE =          queue
;Base =           base
;Limit =          limit
;MinQuant =       min
;MaxQuant =       max
;Boost =          {DECAY | OSCILLATE}
;Timeslice =      tslice
;MinCpuPct =      min percentage
;MaxCpuPct =      max percentage
```

Multiple specifications are permitted within a particular membership criteria (with commas as delimiters), and each criteria need not be specified (unspecified criterias are assumed matches). Although a minimum of one criteria is required. An "&" or Return may be used to indicate the continuation of a specification onto a new line.

The example above shows each parameter on a new line. However, the entire workgroup specification may reside in one physical record. The only requirement is that it is illegal to have two workgroup specification in the same physical record.

Specifications my be "commented out" by using the COMMENT keyword, as shown below. Characters appearing on the same line and after the COMMENT keyword will be ignored.

```
COMMENT   Workgroup =     Old_Finance_Wg
COMMENT   ;QUEUE =         ES
COMMENT   ;Base =          200
COMMENT   ;Limit =         230
COMMENT   ;MinQuant =      200
COMMENT   ;MaxQuant =      1000
COMMENT   ;Boost =         DECAY
COMMENT   ;Timeslice =     400
```

Below is an example of a file that defines workgroups:

```
WORKGROUP=Program_Development
;MEMB_PROGRAM=(EDITOR.PUB.SYS; QEDIT.@.@; HPEDIT.@.@)
;MEMB_LOGON=(MORNING,@.TEST; @.MYTEST)
;BASE=160
;LIMIT=170

WORKGROUP=Payroll_Online
;MEMB_PROGRAM=(PAYROLL.@.PRAPP)
;QUEUE=CS
;BASE=152
;LIMIT=200
;BOOST=OSCILLATE

WORKGROUP=Payroll_Batch
;MEMB_PROGRAM=(PAYROLL.@.PRAPP)
;QUEUE=(DS,ES)
;BASE=180
;LIMIT=230

WORKGROUP=CS_Default
;MEMB_QUEUE=(CS)

WORKGROUP=DS_Default
;MEMB_QUEUE=(DS)

WORKGROUP=ES_Default
;QUEUE=(ES)
```

This file results in following distribution of processes to the workgroups.

Program	User Logon	Queue	Workgroup
EDITOR.PUB.SYS	CHUCK.TEST	CS	Program_Development
EDITOR.PUB.MYTEST	DOUG.MYTEST	CS	CS_Default
EDITOR.PUB.SYS	SLC.TEST	BS	Program_Development
HPEDIT.PUB.SYS	SLC.MYTEST	BS	Program_Development
QEDIT.PUB.SYS	SLC.MYTEST	BS	Program_Development
PAYROLL.PUB.PRAPP	SUSAN.PRAPP	CS	Payroll_Online
PAYROLL.RPT.PRAPP	FRED.PRAPP	DS	Payroll_Batch

Workgroup Information Item Descriptions

The following table provides detailed descriptions of item numbers and corresponding items associated with AIFWGREPLACE

Table 3-49. AIFWGREPLACE Item Descriptions

Item Number	Item Name, Data Type, and Description
19501	**Output Buffer (REC);** Returns the pointer to the buffer in the file where the CI error occured. Record type: buffer_type. The maximum size of n which is user-defined is 512.
19502	**Error Column Number (I32);** Returns the column number where CI error occured.
19503	**CI Error (I32);** Returns the CI error that occured.
19506	**Validate? (B);** Passes an option whereby the data passed is checked for syntax and semantic errors and status is returned without any action being taken on the current population of workgroups. Default : False

A

AIF Status Messages

Item Status Messages

-1	MESSAGE	Read probe failed.
	CAUSE	Caller does not have read access to a virtual address.
	ACTION	Check for uninitialized pointers.
-2	MESSAGE	Write probe failed.
	CAUSE	Caller does not have write access to a virtual address.
	ACTION	Check for uninitialized pointers.
-3	MESSAGE	Read nonscalar probe failed.
	CAUSE	Caller does not have read access to a series of pages in VSM.
	ACTION	Check for uninitialized pointers and counts.
-4	MESSAGE	Write nonscalar probe failed.
	CAUSE	Caller does not have write access to a series of pages in VSM.
	ACTION	Check for uninitialized pointers and counts.
-5	MESSAGE	Bad pointer was encountered.
	CAUSE	The address is uninitialized.
	ACTION	Check for uninitialized pointers.
-6	MESSAGE	Badly aligned pointer was encountered.
	CAUSE	The pointer is incorrectly aligned.
	ACTION	Check for uninitialized pointers and alignment requirements.
-7	MESSAGE	A value mismatch was encountered.
	CAUSE	The value passed in ver. array is not the same as int. value.
	ACTION	Call an AIFGET procedure to obtain the correct value.

-8	MESSAGE	Array overflow.
	CAUSE	The dynamic length array passed in was too small to hold all values.
	ACTION	Use AIFSCGET to obtain upper bounds on array sizes required.
-9	MESSAGE	Internal Error. Unable to lock data that needed to be accessed.
	CAUSE	An unexpected error occurred because another process had the data structures locked.
	ACTION	Change the logic of application to check for this error and restart AIF call.

Overall Status Messages

-20	MESSAGE	Verification arrays wrongly specified.
	CAUSE	Verification arrays to PUT were incorrectly specified.
	ACTION	Pass all 3 or none.
-21	MESSAGE	Bad overall status.
	CAUSE	The overall status was inaccessible for write access.
	ACTION	Check for uninitialized pointers.
-22	MESSAGE	Bad item status.
	CAUSE	The item status was inaccessible for write access.
	ACTION	Check for uninitialized pointers.
-23	MESSAGE	Bad verification item status.
	CAUSE	The verification item status was inaccessible for write access.
	ACTION	Check for uninitialized pointers.
-24	MESSAGE	Verification failed.
	CAUSE	The verification for PUT failed.
	ACTION	Check the ver item statuses for more information.
-25	MESSAGE	Incorrect user capability.
	CAUSE	The AIF procedure called is inaccessible with the specified user id.
	ACTION	Purchase the referenced AIF product component.
-26	MESSAGE	Non-existent AIF user ID.
	CAUSE	The user-id specified does not exist.
	ACTION	Use the user ID distributed when AIFs were purchased.
-27	MESSAGE	Invalid search key.
	CAUSE	The `AIFSYSWIDEGET` search key is no longer valid.
	ACTION	Restart `AIFSYSWIDEGET` calls.

-28	MESSAGE	Invalid JSNum.
	CAUSE	The job/session specified does not exist.
	ACTION	Verify if the JSNum exists.
-29	MESSAGE	PID PIN mismatch encountered.
	CAUSE	The PID process has died and a new process with same PIN was born.
	ACTION	Check the PID and the PIN.
-30	MESSAGE	Process has died.
	CAUSE	No process with this PIN exists on the system.
	ACTION	Check the PIN.
-31	MESSAGE	The process is not of type user, or son, or CI.
	CAUSE	Attempt to PUT to a process of a type that is neither user nor son.
	ACTION	Check the process type and the PIN/PID.
-32	MESSAGE	Invalid accounting name.
	CAUSE	**AIFACCTGET/PUT** could not find the specified account name.
	ACTION	Verify the existence of the specified user, group, and account.
-33	MESSAGE	Invalid Fnum for this process.
	CAUSE	The process does not have a file open with this Fnum.
	ACTION	Check the PID - Fnum combination. Verify that the file is not one of the unsupported types. Unsupported file types include: sockets, remote files, null files, dummy files opened for KSAM, and dummy files opened for datacomm.
-34	MESSAGE	A device file was encountered where it is not supposed to be.
	CAUSE	Attempted to **AIFxxGET** or **AIFxxPUT** to a file of device type.
	ACTION	Check the file type and the Fnum-PID combination.
-35	MESSAGE	The UFID does not correspond to the file specified.
	CAUSE	The Fnum was closed and a new file was opened with same fnum.
	ACTION	Check the list of open files using **AIFPROCGET**.

-36	MESSAGE	Not a user file.
	CAUSE	Attempted to PUT to a file with designator, not user.
	ACTION	Check the file designator.
-37	MESSAGE	A directory object was encountered.
	CAUSE	Attempted to PUT to a file that is actually a Dir. object.
	ACTION	Check for Dir Obj. in AIFFILEGET.
-38	MESSAGE	Parameter 1 was badly aligned.
	CAUSE	Parameter 1 was not aligned on a word (4-byte) boundary.
	ACTION	Check for uninitialized pointers.
-39	MESSAGE	Parameter 2 was badly aligned.
	CAUSE	Parameter 2 was not aligned on a word (4-byte) boundary.
	ACTION	Check for uninitialized pointers.
-40	MESSAGE	Parameter 3 was badly aligned.
	CAUSE	Parameter 3 was not aligned on a word (4-byte) boundary.
	ACTION	Check for uninitialized pointers.
-41	MESSAGE	Parameter 4 was badly aligned.
	CAUSE	Parameter 4 was not aligned on a word (4-byte) boundary.
	ACTION	Check for uninitialized pointers.
-42	MESSAGE	Invalid UFID.
	CAUSE	The UFID parameter specified does not exist.
	ACTION	Verify the UFID used.
-43	MESSAGE	Invalid file name.
	CAUSE	The file name specified does not exist.
	ACTION	Verify if the file name exists.

-45	MESSAGE	Return array1 write probe failed.
	CAUSE	User does not have write access to the array passed in.
	ACTION	Check for uninitialized pointers and *num_array_entry*.
-46	MESSAGE	Return array2 write probe failed.
	CAUSE	User does not have write access to the array passed in.
	ACTION	Check for uninitialized pointers and *num_array_entry*.
-47	MESSAGE	Invalid AIF key.
	CAUSE	**AIFSYSWIDEGET** did not recognize the *aif_area* key.
	ACTION	Try 1000, 2000, 5000, 6000, 8000, or 11000.
-48	MESSAGE	Creation of shareable object failed.
	CAUSE	Call to **AIFGLOBACQ** was unsuccessful.
	ACTION	Possibly out of transient disk space.
-49	MESSAGE	Release of shareable object failed.
	CAUSE	Call to **AIFGLOBREL** was unsuccessful.
	ACTION	Verify that the object pointer is valid.
-50	MESSAGE	Missing criteria arrays.
	CAUSE	**AIFSYSWIDEGET** AIF key specified requires criteria arrays.
	ACTION	Use *itemnum_array*, *item_array*, *item_status_array parameters*.
-51	MESSAGE	Bad pointer was encountered for parameter 1.
	CAUSE	The address passed was inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-52	MESSAGE	Bad pointer was encountered for parameter 2.
	CAUSE	The address passed was inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.

-53	MESSAGE	Bad pointer was encountered for parameter 3.
	CAUSE	The address passed was inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-54	MESSAGE	Bad pointer was encountered for parameter 4.
	CAUSE	The address passed was inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-55	MESSAGE	AIFCLOSE failed.
	CAUSE	Either a bad *file_number* was specified, another file with the same name already exists, an illegal disposition (5, 6, 7) exists, or any outstanding write I/Os may have failed.
	ACTION	Use FCHECK to determine why AIFCLOSE failed.
-56	MESSAGE	The address passed for the verification item number array is not accessible to the caller.
	CAUSE	The address passed is inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-57	MESSAGE	The address passed for the verification items array is not accessible to the caller.
	CAUSE	The address passed is inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-58	MESSAGE	The address is not properly aligned for the verification item number to be accessed.
	CAUSE	The address is not properly aligned for the data type to be accessed.
	ACTION	Pass only a variable that has the proper data alignment.
-59	MESSAGE	The address is not properly aligned for the verification items to be accessed.
	CAUSE	The address is not properly aligned for the data type to be accessed.
	ACTION	Pass only a variable that has the proper data alignment.
-60	MESSAGE	The address is not properly aligned for the verification item statuses.
	CAUSE	The address is not properly aligned for the data type to be accessed.
	ACTION	Pass only a variable that has the proper data alignment.

-61	MESSAGE	Unable to access the file AIFKUF.PUB.SYS.
	CAUSE	The file is deleted, or there is not enough disk space to create it.
	ACTION	Create enough disk space, if needed. Reboot the machine.
-63	MESSAGE	Parameter 5 was badly aligned.
	CAUSE	Parameter 5 was not aligned on a word (4-byte boundary).
	ACTION	Check for proper alignment before calling the AIF.
-64	MESSAGE	Parameter 5 not accessible to caller.
	CAUSE	The address passed was inaccessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-65	MESSAGE	Invalid Path Identifier.
	CAUSE	The Path id parameter specified is not valid.
	ACTION	Verify the Path id used.
-66	MESSAGE	Invalid pathname.
	CAUSE	The pathname specified does not exist in the directory.
	ACTION	Verify the pathname used.
-67	MESSAGE	Could not get the Current Working Directory file pointer.
	CAUSE	The CWD file is closed.
	ACTION	Check your application to make sure that you are not closing the CWD file.
-68	MESSAGE	The pathname is too large for the buffer size specified.
	CAUSE	The user defined a buffer which is too small to hold the pathname.
	ACTION	Increase the buffer size.
-69	MESSAGE	Path passed is empty; first character is a terminator or path length is 0.
	CAUSE	The user passed in a pathname which is empty.
	ACTION	Check the application.

-70	MESSAGE	Cannot traverse the directory; a directory file has been opened exclusively.
	CAUSE	A directory is opened exclusively, which is preventing directory traversal.
	ACTION	Re-run the application when the directory file has been closed.
-71	MESSAGE	Incorrect pathname syntax.
	CAUSE	The user has specified a pathname which is not a valid syntax.
	ACTION	Consult the *MPE/iX Commands Reference Manual Volumes 1 and 2* (32650-90003 and 32650-90364) for a description of a valid pathname syntax.
-72	MESSAGE	User/process lacks Traverse Directory permission on a directory component.
	CAUSE	The user/process is lacking one of the required directory permissions.
	ACTION	Assign the user the appropriate directory security access rights.
-73	MESSAGE	User/process lacks Create Directory permission on the parent directory.
	CAUSE	The user/process is lacking one of the required directory permissions.
	ACTION	Assign the user the appropriate directory security access rights.
-74	MESSAGE	User/process lacks Delete Directory permission on the parent directory.
	CAUSE	The user/process is lacking the required directory permission.
	ACTION	Assign the user the appropriate directory security access rights.
-75	MESSAGE	User/process lacks Read Directory permission on the parent directory.
	CAUSE	The user/process is lacking the required directory permission.
	ACTION	Assign the user the appropriate directory security access rights.
-76	MESSAGE	Could not open the HPUID.PUB.SYS file.
	CAUSE	HPUID.PUB.SYS may not exist, may be corrupt, or may be opened exclusively.
	ACTION	If HPUID.PUB.SYS does not exist, create it with the PXUTIL.PUB.SYS utility.
-77	MESSAGE	Could not open the HPGID.PUB.SYS file.
	CAUSE	HPGID.PUB.SYS may not exist, may be corrupt, or may be opened exclusively.
	ACTION	If HPGID.PUB.SYS does not exist, create it with the PXUTIL.PUB.SYS utility.

-78	MESSAGE	Could not retrieve the user entry from the HPUID.PUB.SYS file.
	CAUSE	The user entry could not be found in the HPUID.PUB.SYS file.
	ACTION	Update the HPUID.PUB.SYS file with the PXUTIL.PUB.SYS utility.
-79	MESSAGE	Could not retrieve the group entry from the HPGID.PUB.SYS file.
	CAUSE	The group entry could not be found in the HPGID.PUB.SYS file.
	ACTION	Update the HPGID.PUB.SYS file with the PXUTIL.PUB.SYS utility.
-81	MESSAGE	Cannot return a pathname which is larger than the maximum path size.
	CAUSE	Currently, it is possible to create a pathname too large to return.
	ACTION	Change to the application to return the long pathname relative to your current working directory.
-82	MESSAGE	Can't open a directory during directory traversal due to too many files open.
	CAUSE	Too many files are already open; failed to open a directory.
	ACTION	Check the application.
-83	MESSAGE	Security violation during directory traversal; failed to open directory file.
	CAUSE	Encountered security violation when trying to open a directory file.
	ACTION	Check the user/application security.
-84	MESSAGE	Read probe failed on pathname item key.
	CAUSE	When probing the pathname item key, an error was returned.
	ACTION	Make sure you are not passing in a bad length in the pathname item key.
-85	MESSAGE	The tempfile parm is not valid in conjunction with the pathname item key.
	CAUSE	Temporary files are currently not supported in the Hierarchical File System.
	ACTION	Use the `filename` or `UFID` parameter when interested in a temporary file.
-86	MESSAGE	The pathname length specified is bad.
	CAUSE	User specified a pathname length less than zero or greater than the maximum pathname length.
	ACTION	Check the application.

-89	MESSAGE	Error occurred when trying to get file label for this file. UFID may be bad.
	CAUSE	The UFID passed to AIFSYSWIDEGET may be bad.
	ACTION	Check to see if the file exists.
-90	MESSAGE	An error occurred while trying to obtain ownershop of the device.
	CAUSE	The device is not currently available for use.
	ACTION	Check device. Contact Hewlett-Packard for support.
-91	MESSAGE	An error occurred while trying to release device ownership.
	CAUSE	An unexpected internal error occurred.
	ACTION	Contact Hewlett-Packard for support.
-101	MESSAGE	Unsupported option.
	CAUSE	Port manage access was requested, but is not supported.
	ACTION	Do not attempt to open a port for Port Manager access.
-102	MESSAGE	Too many receive opens on the AIF ports.
	CAUSE	The maximum number of receive opens have already been done.
	ACTION	Check the logic of your application. The maximum is very large.
-103	MESSAGE	Too many opens for manage access on the AIF ports.
	CAUSE	The maximum number of manage opens have already been done.
	ACTION	Check the logic of your application.
-104	MESSAGE	Too many opens for send access on the AIF ports.
	CAUSE	The maximum number of send opens have already been done.
	ACTION	Check the logic of your application. The maximum is very large.
-105	MESSAGE	Invalid ACCESS MODE specified.
	CAUSE	The access mode code is not one of the allowed values.
	ACTION	Check the logic of your application.
-106	MESSAGE	Message length is negative or greater than maximum.
	CAUSE	The specified message length is not valid.
	ACTION	Check the logic of your application.

-107	MESSAGE	Specified port ID is not valid.
	CAUSE	The Port is either not open or is invalid.
	ACTION	Check the logic of your application.
-108	MESSAGE	Attempted to send a message on Port not opened for Send access.
	CAUSE	The calling process does not have the Port open for Send access.
	ACTION	Check the logic of your application.
-109	MESSAGE	Attempted to receive a message from Port not open for Receive access.
	CAUSE	The calling process does not have the Port open for Receive access.
	ACTION	Check the logic of your application.
-110	MESSAGE	Attempted to manage a Port not open for Manage access.
	CAUSE	The calling process is not the Port Manager.
	ACTION	Check the logic of your application.
-111	MESSAGE	A timeout occurred.
	CAUSE	The specified number of seconds has passed.
	ACTION	Verify that the timeout value specified is sufficient.
-112	MESSAGE	No Ports open for receive access, multi-port receive failed.
	CAUSE	The calling process has no ports open for receive access.
	ACTION	Check the logic of your application.
-113	MESSAGE	Attempt to open Port for same access multiple times.
	CAUSE	Process attempted to open same port for same access multiple times.
	ACTION	Check the logic of your application.
-114	MESSAGE	Unsupported procedure.
	CAUSE	A procedure that is not yet supported was called.
	ACTION	Check the logic of your application.
-115	MESSAGE	No zero element terminator was found in the itemnums array.
	CAUSE	No terminator was found in the itemnums array.
	ACTION	Check the logic of your application.

-116	MESSAGE	Invalid password.
	CAUSE	The named port exists, but the password supplied does not match.
	ACTION	Check the logic of your application.
-117	MESSAGE	Internal error.
	CAUSE	The port does not exist.
	ACTION	Perform a dump. Contact Hewlett-Packard for support.
-125	MESSAGE	Itemnums, Items, and Itemstatus not specified together.
	CAUSE	Must pass item option arrays as a triple. All or none.
	ACTION	Check the logic of your application.
-126	MESSAGE	Must complete two-part receive before receive from another port.
	CAUSE	Receive from second port before doing second part of two part receive.
	ACTION	Check the logic of your application.
-127	MESSAGE	Must request message return on 2nd receive of two-part receive.
	CAUSE	The second receive of a two-part receive did not request message.
	ACTION	Check the logic of your application.
-128	MESSAGE	The message length specified on send was larger than the max length specified when the Port was created.
	CAUSE	Message cannot be larger than the specified size.
	ACTION	Check the logic of your application.
-129	MESSAGE	A NoWait receive was done while there was no message ready.
	CAUSE	A NoWait receive was done while there was no message ready.
	ACTION	Check the logic of your application.
-130	MESSAGE	An attempt was made to access a port which is not open for the calling process.
	CAUSE	The port was not opened by the calling process.
	ACTION	Check the logic of your application.

-900 thru -942	MESSAGE	Internal Error.
	CAUSE	An unexpected internal error occurred.
	ACTION	Contact Hewlett-Packard for support.
-943	MESSAGE	Internal Error.
	CAUSE	Security Internal error returned from the FS_SEC_ACCESS routine.
	ACTION	Contact Hewlett-Packard for support.
-944	MESSAGE	Internal Error.
	CAUSE	An unexpected error occurred from the hierarchical directory routines.
	ACTION	Contact Hewlett-Packard for support.
-945	MESSAGE	Internal Error.
	CAUSE	An unexpected error occurred from the directory traversal routines.
	ACTION	Contact Hewlett-Packard for support.
-946	MESSAGE	Internal error. Error returned from HPDIRREAD when reading a directory file.
	CAUSE	An unexpected error occurred from the directory traversal routines.
	ACTION	Contact Hewlett-Packard for support.
-947	MESSAGE	Internal error. HPFOPEN returned bad status when opening a directory file.
	CAUSE	An unexpected error occurred from the directory traversal routines.
	ACTION	Contact Hewlett-Packard for support.
-948	MESSAGE	Internal error. The directory UFID is bad.
	CAUSE	An unexpected error occurred from the directory traversal routines.
	ACTION	Contact Hewlett-Packard for support.
-949 thru -999	MESSAGE	Internal Error.
	CAUSE	An unexpected internal error occurred.
	ACTION	Contact Hewlett-Packard for support.

Job/Session Information Status Messages

-1001	MESSAGE	Invalid item number passed in *itemnum_array* to AIFJSGET.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-1002	MESSAGE	Invalid item number passed in *ver_item_nums* to AIFJSPUT.
	CAUSE	A non-zero, invalid item number was passed in *ver_item_nums*.
	ACTION	Pass an appropriate value and end with a zero.
-1003	MESSAGE	Invalid item number passed in *itemnum_array* to AIFJSPUT.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-1004	MESSAGE	Job not in WAIT, SCHED, or EXEC* state.
	CAUSE	The job indicated was not in WAIT, SCHED, or EXEC* state.
	ACTION	Call AIFSYSWIDEGET for a list of jobs in WAIT, SCHED, or EXEC* states.
-1005	MESSAGE	Input Priority not in the range 0-15.
	CAUSE	The input priority value was not in the range 0-15.
	ACTION	Use a value from 0 to 15 when setting the input priority.
-1006	MESSAGE	Unable to change input priority or output device.
	CAUSE	There is an unexpected system problem.
	ACTION	Contact your Response Center.
-1007	MESSAGE	Not a job.
	CAUSE	The input key given was not for a job.
	ACTION	Call AIFSYSWIDEGET for a list of jobs.

-1008	MESSAGE	Output Priority not in the range 0-14.
	CAUSE	The output priority value was not in the range 0-14.
	ACTION	Use a value from 0 to 14 when setting the output priority.
-1009	MESSAGE	Executing Priority not equal to 100, 150, 200, or 250.
	CAUSE	The executing priority value was not equal to 100, 150, 200, or 250.
	ACTION	Use a value equal to 100, 150, 200, or 250 for executing priority.
-1010	MESSAGE	CPU Limit was not in the range -1 to 32767.
	CAUSE	CPU Limit value was not in the range -1 to 32767.
	ACTION	Use a value from 1 to 32767, or -1 to indicate unlimited CPU.
-1011	MESSAGE	Job/Session not in right state.
	CAUSE	Information requested is not accessible in job/session's current state.
	ACTION	Check item description for state limitations.
-1012	MESSAGE	Internal Error.
	CAUSE	Could not get the Avesta address.
	ACTION	Contact Hewlett-Packard for support.
-1013	MESSAGE	Internal Error. Job/session not in correct state.
	CAUSE	Job/session `jskey` is zero.
	ACTION	Contact Hewlett-Packard for support.

Process Information Status Messages

-2001	MESSAGE	Invalid item number passed in itemnum_array to AIFPROCGET.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-2002	MESSAGE	Invalid item number passed in itemnum_array to AIFPROCPUT.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-2003	MESSAGE	Invalid item number passed in itemnum_array to AIFPROCPUT.
	CAUSE	A non-zero invalid item number was passed in ver_item_nums.
	ACTION	Pass an appropriate value and end with a zero.
-2005	MESSAGE	Invalid CM Error.
	CAUSE	CM error should be in the range of a shortint.
	ACTION	Check for uninitialized pointers and counts and correct range.
-2006	MESSAGE	Invalid CM error index.
	CAUSE	CM error should be in the range of 1..6.
	ACTION	Check for uninitialized pointers and counts and correct range.
-2007	MESSAGE	Invalid NM Error Index.
	CAUSE	NM error should be in the range of 0..16.
	ACTION	Check for uninitialized pointers and counts and correct range.
-2008	MESSAGE	Invalid Scheduling Queue value.
	CAUSE	Sched. Queue should be in the range 0..4.
	ACTION	Check for uninitialized pointers and counts and correct range.
-2009	MESSAGE	Invalid MPE/iX priority value.
	CAUSE	CM error should be in the range of 1..MaxShortint.
	ACTION	Check for uninitialized pointers and counts and correct range.

-2010	MESSAGE	Priority - scheduling class mismatch.
	CAUSE	Priority and scheduling class should match the global limits.
	ACTION	Check for uninitialized pointers and counts and correct range.
-2011	MESSAGE	Invalid capability mask.
	CAUSE	Attempt was made to change to invalid capability mask.
	ACTION	Capability mask must have only lower 16 bits turned on.
-2012	MESSAGE	Invalid general resources mask.
	CAUSE	Attempt was made to change to invalid resource mask.
	ACTION	Resource mark must have only lower 16 bits turned on.
-2013	MESSAGE	The value specified is not in the allowed range.
	CAUSE	Application attempted to do a PUT with a value out of range.
	ACTION	Correct the application.

AIFCHANGELOGON Status Messages

-2501	MESSAGE	*Logon_string* or *logon_desc parameter* required.
	CAUSE	Caller must specify either the *logon_cmd* parameter or the *logon_desc* parameter.
	ACTION	Change the call to supply either the *logon_cmd*, *logon_desc* parameter, or both.
-2502	MESSAGE	Probe failed (*logon_desc*).
	CAUSE	Caller does not have write access to *logon_desc*.
	ACTION	Check call. You do not have write access to the address passed for the *logon_desc* parameter.
-2503	MESSAGE	Probe failed (*logon_cmd*).
	CAUSE	Caller does not have read access to *logon_cmd*.
	ACTION	Check call. You do not have read access to the address passed for the *logon_cmd* parameter.
-2504	MESSAGE	Probe failed (*error_status*).
	CAUSE	Caller does not have write access to *error_status*.
	ACTION	Check call. You do not have write access to the address passed for the *error_status* parameter.
-2510	MESSAGE	Syntax error (*logon_cmd*).
	CAUSE	*Logon_string* parameter contained a syntax error. If the caller passed the *error_status* parameter, then the scanner/parser status will be returned in that parameter.
	ACTION	Pass a syntactically valid *logon_cmd*. To print a more specific syntax error, pass the *error_status* parameter to **AIFCHANGELOGON**. The value returned from **AIFCHANGELOGON** can be passed to the HPERRMSG intrinsic.

-2511	MESSAGE	No termination character (*logon_cmd*).
	CAUSE	The caller did not terminate the logon string with a valid string terminator character. (Either a NUL or a CR is required).
	ACTION	Check call. You must terminate the *logon_cmd* parameter with either a NUL character (value=0) or a carriage return (value=13). Note that the maximum length for the *logon_cmd* parameter is 256 bytes. If you passed a *logon_cmd* longer than 256 bytes and the terminator is beyond the 256th byte or not present at all, **AIFCHANGELOGON** returns this error.
-2515	MESSAGE	Non-existent account.
	CAUSE	The account specified as the target does not exist.
	ACTION	Specify a target account that exists on your system.
-2516	MESSAGE	Non-existent user.
	CAUSE	The user specified as the target does not exist.
	ACTION	Specify a target user that exists on your system (within the target account specified).
-2517	MESSAGE	Non-existent group.
	CAUSE	The group specified as the target does not exist.
	ACTION	Specify a target group that exists on your system (within the target account specified).
-2518	MESSAGE	Invalid home group in directory.
	CAUSE	No group was specified, so the target user's home group was used. The user's home group does not exist.
	ACTION	Do not default the group for this user, because the home group specified for this user does not exist on your system (Eg: it has been deleted).
-2519	MESSAGE	No home group for user in directory.
	CAUSE	No group was specified and the user does not have a home group.
	ACTION	Do not default the group for this user, because this user has no home group (Eg: none was specified on the :NEWUSER command when this user was created).

-2520	MESSAGE	Invalid account password specified.
	CAUSE	An invalid account password was specified.
	ACTION	You specified an invalid password for the target account. Specify the correct password.
-2521	MESSAGE	Invalid user password specified.
	CAUSE	An invalid user password was specified.
	ACTION	You specified an invalid password for the target user. Specify the correct password.
-2522	MESSAGE	Invalid group password specified.
	CAUSE	An invalid group password was specified.
	ACTION	You specified an invalid password for the target group. Specify the correct password.
+2523	MESSAGE	Unnecessary account password specified. WARNING only.
	CAUSE	A password was specified for the account, but the account does not have a password. The extra password was ignored.
	ACTION	To avoid the warning, do not pass an account password for this target account.
+2524	MESSAGE	Unnecessary user password specified. WARNING only.
	CAUSE	A password was specified for the user, but the user does not have a password. The extra password was ignored.
	ACTION	To avoid the warning, do not pass a user password for this target user.
+2525	MESSAGE	Unnecessary group password specified. WARNING only.
	CAUSE	A password was specified for the group, but the group does not have a password. The extra password was ignored.
	ACTION	To avoid the warning, do not pass a group password for this target group.
+2526	MESSAGE	A password aging warning is in effect. WARNING only.
	CAUSE	A user password warning is set by the MPE/iX Security Monitor.
	ACTION	The user password is about to expire. The user password must be replaced or it will expire. Contact your System Manager for further assistance.

-2527	MESSAGE	The user password has expired.
	CAUSE	The user password expiration is set by the MPE/iX Security Monitor.
	ACTION	The user has an expired password which must be replaced. Contact your System Manager for further assistance.
-2528	MESSAGE	The user password is invalid.
	CAUSE	The user password has exceeded the maximum lifetime allowed by the MPE/iX Security Monitor.
	ACTION	The user password is invalid see your System Manager for further assistance.
-2529	MESSAGE	The user is disabled.
	CAUSE	A threat detection violation was encountered by the MPE/iX Security Monitor.
	ACTION	The user is disabled, see your System Manager for further assistance.
-2541	MESSAGE	Non-existent target user, group, or account.
	CAUSE	This error should only occur if the user, account, or group is purged after **AIFCHANGELOGON** has verified they exist (and that you have specified the correct passwords).
	ACTION	Treat this error the same as if the target user, account, or group does not exist on your system. (See messages -2515, -2516, or -2517.)
-2550	MESSAGE	Internal Error.
	CAUSE	JSINFO returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.

-2551	MESSAGE	Internal Error.
	CAUSE	JSSET returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2552	MESSAGE	Internal Error.
	CAUSE	DIRECLOGOFF returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2553	MESSAGE	Internal Error.
	CAUSE	CM_SCHANGE_XDD returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2554	MESSAGE	Internal Error.
	CAUSE	RELSIR returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2555	MESSAGE	Internal Error.
	CAUSE	GETSIR returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2556	MESSAGE	Internal Error.
	CAUSE	CONVERT_DST returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2557	MESSAGE	Internal Error.
	CAUSE	Unexpected ESCAPE from a subsystem called by **AIFCHANGELOGON**.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.

-2560	MESSAGE	Internal Error.
	CAUSE	COREPARSER returned a token longer then the maximum token length that was specified to it.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2561	MESSAGE	Internal Error.
	CAUSE	An invalid case occurred in a case statement in BUILD_LOGON_DESC.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2562	MESSAGE	Internal Error.
	CAUSE	An invalid case occurred in a case statement in CHECK_DIRECTORY.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2563	MESSAGE	Internal Error.
	CAUSE	An invalid case occurred in a case statement in CHECK_PASSWORD.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-2564	MESSAGE	Internal error.
	CAUSE	GETDATASEG returned bad status.
	ACTION	Contact your Hewlett-Packard support representative.
-2565	MESSAGE	Internal error.
	CAUSE	RELDATASEG returned bad status.
	ACTION	Contact your Hewlett-Packard support representative.

-2566	MESSAGE	Internal error.
	CAUSE	Rebuild of the temporary directory failed.
	ACTION	Contact your Hewlett-Packard support representative.
-2567	MESSAGE	Internal error.
	CAUSE	CM_BUILD_JDT returned bad status.
	ACTION	Contact your Hewlett-Packard support representative.
-2568	MESSAGE	Rebuilding the temporory directory failed, the application has open temporary files.
	CAUSE	Temporary files must be closed prior to calling AIFCHANGELOGON unless the option to keep the temporary directory is specified.
	ACTION	Check the logic of the application.
-2569	MESSAGE	Internal error.
	CAUSE	Adjust user count error returned.
	ACTION	Contact your Hewlett-Packard support representative.
-2570	MESSAGE	Internal error.
	CAUSE	Process could not be moved to the new workgroup.
	ACTION	Contact your Hewlett-Packard support representative.
-2571	MESSAGE	Internal error.
	CAUSE	Attempt to duplicate the file descriptor failed.
	ACTION	Contact your Hewlett-Packard support representative.

-2572	MESSAGE	Internal error.
	CAUSE	Unexpected entry found in temporary directory.
	ACTION	Contact your Hewlett-Packard support representative.
-2573	MESSAGE	Internal error.
	CAUSE	Attempt to read temporary directory entry failed.
	ACTION	Contact your Hewlett-Packard support representative.
-2574	MESSAGE	Internal error.
	CAUSE	Attempt to link entry in temporary directory failed.
	ACTION	Contact your Hewlett-Packard support representative.

System Configuration Status Messages

-3001	MESSAGE	Invalid item number passed in *itemnum_array* to AIFSCGET.
	CAUSE	A non-zero invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-3002	MESSAGE	Invalid item number passed in *vernum_array* to AIFSCPUT.
	CAUSE	A non-zero, invalid item number was passed in *vernum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-3003	MESSAGE	Invalid item number passed in *itemnum_array* to AISCPUT.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-3004	MESSAGE	Unable to obtain system outfence.
	CAUSE	There is an unexpected system problem.
	ACTION	Contact your Response Center.
-3005	MESSAGE	Job fence not in range 0 - 14.
	CAUSE	The job fence value was not in the range 0 - 14.
	ACTION	Use a value from 0 to 14 when setting the system job fence.
-3006	MESSAGE	Job limit not in range 0 - 16383.
	CAUSE	The job limit value was not in the range 0 - 16383.
	ACTION	Use a value from 0 to 16383 when setting the system job limit.
-3007	MESSAGE	Session limit not in range 0 - 16383.
	CAUSE	The session limit value was not in the range 0 - 16383.
	ACTION	Use a value from 0 to 16383 when setting the system session limit.

-3008	MESSAGE	Next job number not in range 1 - 16383.
	CAUSE	The next job number value was not in the range 1 - 16383.
	ACTION	Use a value from 1 to 16383 when setting the next job number.
-3009	MESSAGE	Next session number not in range 1 - 16383.
	CAUSE	The next session number value was not in the range 1 - 16383.
	ACTION	Use a value from 1 to 16383 when setting the next session number.
-3010	MESSAGE	Job security not equal to 0 or 3.
	CAUSE	The job security value was not equal to 0 or 3.
	ACTION	Use either 0 or 3 when setting the system job security.
-3011	MESSAGE	System outfence not in range 1 - 14.
	CAUSE	The system outfence value was not in the range 1 - 14.
	ACTION	Use a value from 1 to 14 when setting the system outfence.
-3012	MESSAGE	Subqueue value not in range 127 - 13567.
	CAUSE	The subqueue value was not in the range 127 - 13567.
	ACTION	Use a value from 127 to 13567 when setting any subqueue base or limit.
-3013	MESSAGE	Unable to change system outfence.
	CAUSE	There is an unexpected system problem.
	ACTION	Use a value from 127 to 13567 when setting the CS subqueue limit.
-3014	MESSAGE	Invalid value passed in AIFTIME.
	CAUSE	A negative or otherwise invalid value was passed in AIFTIME.
	ACTION	Pass a positive value to AIFTIME.
-3015	MESSAGE	Invalid boost property was specified.
	CAUSE	The boost property specified was not in the valid range.
	ACTION	Pass an appropriate value for the boost range.

-3016	MESSAGE	Internal error.
	CAUSE	Unexpected error occurred when attempting to update dispatcher items.
	ACTION	Contact Hewlett-Packard for support.
-3017	MESSAGE	Write to PDC failed.
	CAUSE	There is an unexpected system problem.
	ACTION	Contact Hewlett-Packard for support.
-3018	MESSAGE	Read of PDC failed.
	CAUSE	There is an unexpected system problem.
	ACTION	Contact Hewlett-Packard for support.
-3019	MESSAGE	Internal Error.
	CAUSE	Could not freeze PDC buffer.
	ACTION	Contact Hewlett-Packard for support.
-3020	MESSAGE	Internal Error.
	CAUSE	Could not unfreeze PDC buffer.
	ACTION	Contact Hewlett-Packard for support.
-3021	MESSAGE	Unable to return global security information.
	CAUSE	The security information was not found.
	ACTION	The HP 3000 Security Monitor/iX is not found. Check with the system manager.
-3023	MESSAGE	An illegal value was specified for the lower limit.
	CAUSE	The lower limit of the job, session, input spoolid, or output spoolid range must be a positive integer. If the upper limit is 0 (meaning that the system should use the absolute limit of the counter as an upper limit), then the lower limit must be less than that absolute limit. If the upper limit is non-zero, the lower limit must be less than the upper limit.
	ACTION	Use a value in the correct range. If you are not PUTting the upper limit in the same AIF call, it may be necessary to obtain the current upper limit via AIFSCGET.

-3024	MESSAGE	An illegal value was specified for the upper limit.
	CAUSE	The upper limit of the job, session, input spoolid, or output spoolid range must either be 0 (meaning that the system should use the absolute limit of the range) or a positive integer greater than the lower limit but less than the absolute limit of the specified counter.
	ACTION	Use a value in the correct range. If you are not PUTting the lower limit in the same AIF call, it may be necessary to obtain the current lower limit via AIFSCGET.
-3025	MESSAGE	An illegal value was specified for the next input spoolid.
	CAUSE	The value specified either already exists as an input spoolid or is not in the range 1 - 9999999.
	ACTION	Use a value in the correct range and which is not currently in use.
-3026	MESSAGE	An illegal value was specified for the next output spoolid.
	CAUSE	The value specified either already exists as an output spoolid or is not in the range 1 - 9999999.
	ACTION	Use a value in the correct range and which is not currently in use.

Local File Information Status Messages

-4001	MESSAGE	Invalid Local File Get Item Number.
	CAUSE	Invalid Local File Get Item Number.
	ACTION	Check for uninitialized pointers/item numbers.
-4002	MESSAGE	Invalid Local File Put Item Number.
	CAUSE	Invalid Local File Put Item Number.
	ACTION	Check for uninitialized pointers/item numbers.
-4003	MESSAGE	Invalid Local File Verify Item Number .
	CAUSE	Invalid Local File Verify Item Number.
	ACTION	Check for uninitialized pointers/item numbers.
-4004	MESSAGE	A CM file was specified.
	CAUSE	A CM file was specified where a non CM file was required.
	ACTION	Check for item number and the CM file? item.
-4005	MESSAGE	A CM file is required here.
	CAUSE	A non CM file was specified where a CM file was required.
	ACTION	Check for item number and the CM file? item.
-4006	MESSAGE	A NM file is required here.
	CAUSE	A non NM file was specified where a NM file was required.
	ACTION	Check for item number.
-4007	MESSAGE	A file is required here.
	CAUSE	A non file is specified where a file is required.
	ACTION	Check for item number.

-4008	MESSAGE	Invalid record number.
	CAUSE	Record number should be positive.
	ACTION	Check for item number.
-4009	MESSAGE	An invalid offset was specified.
	CAUSE	Record offset should be positive.
	ACTION	Check for item number.
-4010	MESSAGE	Invalid access rights specified.
	CAUSE	File access rights should be in the range 0..255.
	ACTION	Check for item number.
-4011	MESSAGE	Invalid privilege level was specified.
	CAUSE	Privileged level should be 2..3 and prev. level should also be 2..3.
	ACTION	Check for item number.
-4012	MESSAGE	Invalid locking specified.
	CAUSE	Locking should be in the range 0..4.
	ACTION	Check for item number.
-4014	MESSAGE	CM or variable length records file is needed.
	CAUSE	A CM of variable length records file is needed.
	ACTION	Check for item number.
-4015	MESSAGE	A non device file is required.
	CAUSE	A non device file is required.
	ACTION	Check for item number.
-4016	MESSAGE	Invalid block offset specified.
	CAUSE	Block offset should be positive.
	ACTION	Check for item number.
-4017	MESSAGE	Invalid file type was specified.
	CAUSE	A file type was specified where a different file type was required.
	ACTION	Check for item number.

Global Warning Messages

+4500	MESSAGE	The HFS file has been opened by ufid.
	CAUSE	When opened by ufid, it is not always possible to return the pathname.
	ACTION	No action.
+4501	MESSAGE	The file is a HFS file and cannot be represented by an MPE filename.
	CAUSE	MPE filename syntax is not appropriate to represent a HFS file.
	ACTION	Use the pathname items to retrieve the pathname.
+4502	MESSAGE	The MPE file cannot be represented by a HFS pathname.
	CAUSE	Not all MPE files (for example, $TEMPDIRC) can be represented by a HFS pathname.
	ACTION	Use the appropriate MPE filename item.
+4503	MESSAGE	The file was opened with no parent ufid filled in.
	CAUSE	Some MPE files are opened without the parent ufid field filled in.
	ACTION	No action.
+4504	MESSAGE	Cannot retrieve the pathname for this file.
	CAUSE	Cannot always get a pathname for a HFS file which is opened by ufid.
	ACTION	No action.
+4505	MESSAGE	Cannot retrieve the full pathid for this file (i.e. no parent ufid/linkid.)
	CAUSE	Cannot always get a full pathid (e.g. if UFID item key is used for HFS file.) ufid.
	ACTION	Specify the pathid or pathname item key for HFS files.

Global File Information Status Messages

-5001	MESSAGE	Invalid item number passed in *itemnum_array* for AIFFILEGGET.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-5002	MESSAGE	Invalid item number passed in *itemnum_array* for AIFFILEGGET.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-5003	MESSAGE	Invalid item number for global verify.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-5005	MESSAGE	Timestamp is not in the past.
	CAUSE	The date and time passed in is not in the past.
	ACTION	Pass only a date and time that has already passed.
-5006	MESSAGE	Close disposition not in the range 0 - 5.
	CAUSE	The close disposition passed is not in the range 0 - 5.
	ACTION	Pass a number from 0 to 5 when setting the close disposition.
-5007	MESSAGE	No file name or UFID specified.
	CAUSE	A file name or UFID key must be specified.
	ACTION	Specify file name or UFID.
-5008	MESSAGE	File code cannot be changed to a negative value.
	CAUSE	The file code passed in is negative and therefore cannot be changed.
	ACTION	Pass a number >= 0 only when changing a file code.
-5009	MESSAGE	Invalid item number passed in *itemnum_array* for AIFFILEGGET.
	CAUSE	A non-zero invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.

-5010	MESSAGE	Invalid item number for global verify.
	CAUSE	A non-zero invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate value and end with a zero.
-5011	MESSAGE	File specified is privileged.
	CAUSE	Access is not allowed on privileged files.
	ACTION	Do not access privileged files.
-5012	MESSAGE	Put failed to file.
	CAUSE	Transaction management failed.
	ACTION	Contact your response center.
-5013	MESSAGE	Directory is on a read-only volume.
	CAUSE	Cannot alter a directory entry on a read-only volume.
	ACTION	Do not use this item for directory entries on a read-only volume.
-5014	MESSAGE	There are too many levels in the directory.
	CAUSE	The number of levels traversed exceeds the current maximum number.
	ACTION	Set current working directory (CWD) to lower level directory and pass relative pathname.

Accounting Information Status Message

-6001	MESSAGE	Invalid item number passed in *itemnum_array* for AIFACCTGET.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate item number as specified in the reference manual.
-6002	MESSAGE	Invalid item number passed in *itemnum_array* for AIFACCTPUT.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate item number as specified in the reference manual.
-6003	MESSAGE	Invalid item number passed in *ver_itemnum_array* for AIFACCTPUT.
	CAUSE	A non-zero, invalid item number was passed in *ver_itemnum_array*.
	ACTION	Pass an appropriate item number as specified in the reference manual.
-6004	MESSAGE	Item not implemented.
	CAUSE	The item number specified in the *itemnum_array* is not implemented.
	ACTION	In a future release the item will be implemented.
-6005	MESSAGE	Invalid item number for the specified key.
	CAUSE	The item specified is not obtainable with the given *directory_name* key.
	ACTION	Verify that the *directory_name* key specifies the directory object desired.
-6006	MESSAGE	Invalid Password passed.
	CAUSE	The password specified is not the same as for the directory.
	ACTION	Verify that the *directory_name* key specifies the directory object desired.
-6007	MESSAGE	Password cannot be more than 8 characters long.
	CAUSE	A password with more than 8 characters was specified.
	ACTION	Check the password specified in the application.

-6008	MESSAGE	Password must meet minimum length specified by the HP Security Monitor.
	CAUSE	Password is less than the required minimum password length.
	ACTION	Check the password length specified in the application.
-6009	MESSAGE	Password must change.
	CAUSE	The new password input is the same as the old password.
	ACTION	Check the value input in the application.
-6010	MESSAGE	Password is required.
	CAUSE	A HP Security Monitor feature is set requiring a password.
	ACTION	Check the value input in the application.
-6011	MESSAGE	Password can't change.
	CAUSE	A minimum password age feature is set for the HP Security Monitor.
	ACTION	Contact the system manager for assistance.

Spool File and Spooler Process Information Status Messages

-8001	MESSAGE	Unexpected escape.
	CAUSE	Unexpected error occurred in low level code called by NMS AIFs.
	ACTION	Contact Hewlett-Packard for support.
-8002	MESSAGE	Invalid device.
	CAUSE	The device is invalid.
	ACTION	Check the device passed.
-8003	MESSAGE	The device is not spooled.
	CAUSE	The device is not spooled.
	ACTION	Start spool on the device.
-8004	MESSAGE	Invalid data address passed.
	CAUSE	The address passed is not accessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-8005	MESSAGE	Invalid item number passed in *itemnum_array*.
	CAUSE	A non-zero, invalid item number was passed in the *itemnum_array*.
	ACTION	Pass a valid item number.
-8006	MESSAGE	Invalid item value.
	CAUSE	The item value passed is either out of range or illegal.
	ACTION	Pass a valid item value.
-8007	MESSAGE	Invalid AIF user.
	CAUSE	The *user_id* is not valid.
	ACTION	Pass a valid user id or call **AIFACCESSON** before calling this routine.

-8008	MESSAGE	Caller not in privileged mode.
	CAUSE	The process calling this interface is at HW ring level 3.
	ACTION	Call GETPRIVMODE to promote process to HW ring level 2.
-8009	MESSAGE	Cannot get information for the given device.
	CAUSE	Error occurred while interfacing with the native mode device file.
	ACTION	Contact Hewlett-Packard for support.
-8010	MESSAGE	The ldev number passed is not valid.
	CAUSE	Cannot convert the LDEV number to a device name.
	ACTION	Check the LDEV number passed.
-8011	MESSAGE	Cannot delimit the device name passed.
	CAUSE	Error occurred while interfacing with the native mode device file.
	ACTION	Check if the device passed is valid.
-8012	MESSAGE	Cannot lock SPIT.
	CAUSE	Error occurred while trying to lock SPIT.
	ACTION	Check if NMS is running properly.
-8013	MESSAGE	Cannot unlock SPIT.
	CAUSE	Error occurred while trying to unlock SPIT.
	ACTION	Check if NMS is running properly.
-8014	MESSAGE	Cannot find the spooler process for the given device in SPIT.
	CAUSE	The device is not spooled.
	ACTION	Startspool on the device.
-8015	MESSAGE	Failed to verify that the table entry is in the expected state; no PUT is done.
	CAUSE	The table entry is not in the expected state.
	ACTION	Call the GET AIF to obtain the current state of the table entry.
-8017	MESSAGE	Wrong number of device in the given device class.
	CAUSE	NMS internal error.
	ACTION	Submit SR.

-8018	MESSAGE	The device does not exist.
	CAUSE	The device does not exist.
	ACTION	Check if the device passed is valid.
-8219	MESSAGE	Cannot update the device information.
	CAUSE	Error occurred while interfacing with the native mode device file.
	ACTION	Submit an SR.
-8020	MESSAGE	Cannot build spooler internal target queue list.
	CAUSE	Error occurred while building NMS internal target queue list.
	ACTION	Submit an SR.
-8021	MESSAGE	The value passed for *q_state* is invalid.
	CAUSE	The value passed for *q_state* is invalid.
	ACTION	Pass a valid value for *q_state*.
-8022	MESSAGE	The value passed for finishing strategy is invalid.
	CAUSE	The value passed for finishing strategy is invalid.
	ACTION	Pass a valid value for finishing strategy.
-8023	MESSAGE	The value passed for keep/nokeep is invalid.
	CAUSE	The value passed for keep/nokeep is invalid.
	ACTION	Pass a valid value for keep/nokeep.
-8024	MESSAGE	The value passed for *direction* is invalid.
	CAUSE	The value passed for *direction* is invalid.
	ACTION	Pass a valid value for *direction*.
-8025	MESSAGE	Both *keep* and *finish* end_of_copy are passed.
	CAUSE	It is illegal to suspend spool with both *keep* and *finish* end_of_copy.
	ACTION	Fix the procedure call.

-8026	MESSAGE	Both *offset* and *finish* end_of_copy are passed.
	CAUSE	It is illegal to suspend with *finish* end_of_copy and a non-zero offset.
	ACTION	Fix the procedure call.
-8027	MESSAGE	NMS internal error.
	CAUSE	NMS internal error.
	ACTION	Submit SR.
-8028	MESSAGE	Cannot send message to a spooler process.
	CAUSE	Error occurred while trying to send message to a spooler process.
	ACTION	Check if the spooler process is running properly.
-8029	MESSAGE	Cannot retrieve information from SPIT.
	CAUSE	Error occurred while retrieving information from SPIT.
	ACTION	Contact Hewlett-Packard for support.
-8030	MESSAGE	Cannot update information in the SPIT.
	CAUSE	Error occurred while updating SPIT.
	ACTION	Contact Hewlett-Packard for support.
-8031	MESSAGE	The spooler process is in an invalid state.
	CAUSE	NMS internal error.
	ACTION	Contact Hewlett-Packard for support.
-8032	MESSAGE	Cannot lock SPFDIR.
	CAUSE	Error occurred while trying to lock SPFDIR.
	ACTION	Check if NMS is running properly.
-8033	MESSAGE	Cannot unlock SPFDIR.
	CAUSE	Error occurred while trying to unlock SPFDIR.
	ACTION	Check if NMS is running properly.
-8034	MESSAGE	Neither the spool file ID or the UFID is passed.
	CAUSE	This AIF requires that either the spool file ID or the UFID be passed.
	ACTION	Pass the spool file ID or UFID.

-8035	MESSAGE	Cannot convert the spool file ID to UFID.
	CAUSE	Error occurred most likely because the file does not exist.
	ACTION	Check the spool file ID passed.
-8036	MESSAGE	Invalid spool file UFID.
	CAUSE	The given UFID is invalid.
	ACTION	Check the UFID passed.
-8037	MESSAGE	Invalid spool file ID.
	CAUSE	The given spool file ID is invalid.
	ACTION	Check the spool file ID passed.
-8038	MESSAGE	Mismatching spool file ID and UFID.
	CAUSE	The spool file ID and UFID passed do not identify the same spool file.
	ACTION	Check the spool file ID and UFID passed.
-8039	MESSAGE	Cannot find the spool file.
	CAUSE	The spool file does not exist.
	ACTION	Check the spool file ID or UFID passed.
-8040	MESSAGE	Cannot lock the GUFD for the spool file.
	CAUSE	Error occurred while trying to lock the GUFD for the spool file.
	ACTION	Contact Hewlett-Packard for support.
-8041	MESSAGE	Cannot unlock the GUFD for the spool file.
	CAUSE	Error occurred while trying to unlock the GUFD for the spool file.
	ACTION	Contact Hewlett-Packard for support.
-8042	MESSAGE	Invalid item number for an input spool file.
	CAUSE	PUT is not allowed for this item with an input spool file.
	ACTION	Pass a valid item number.

-8043	MESSAGE	The given spool file state value is out of range.
	CAUSE	The value for spool file state is out of range.
	ACTION	Pass a valid spool file state.
-8044	MESSAGE	Cannot modify the state of a :JOB input spool file.
	CAUSE	Cannot modify the state of a :JOB input spool file.
	ACTION	Don't change the state of a :JOB input spool file.
-8045	MESSAGE	Cannot change the state of a :DATA input spool file to other then DEL_PENDING.
	CAUSE	The only valid state change for a :DATA input spool file is to DEL_PENDING.
	ACTION	Don't change the state of a :DATA input spool file to other than DEL_PENDING.
-8046	MESSAGE	The :DATA input spool file is not in READY state.
	CAUSE	Can only change the :DATA input spool file state from READY to DEL_PENDING.
	ACTION	Check the state of the spool file.
-8047	MESSAGE	The given output priority is out of range.
	CAUSE	Output priority must be within 0 to 14.
	ACTION	Pass a valid output priority.
-8048	MESSAGE	The given spool file disposition is out of range.
	CAUSE	The value for spool file disposition must be either 1 or 2.
	ACTION	Pass a valid spool file disposition.
-8049	MESSAGE	The given incomplete flag is out of range.
	CAUSE	The value for the incomplete flag must be either 0 or 1.
	ACTION	Pass a valid incomplete flag.
-8050	MESSAGE	The given value for the number of copies is out of range.
	CAUSE	The range for number of copies is 1 to 65535.
	ACTION	Pass a valid number for the number of copies.

-8051	MESSAGE	Cannot change the date/time of a spool file that is being created.
	CAUSE	The spool file has not yet been closed after being created.
	ACTION	Change the spool file date/time after it becomes READY.
-8052	MESSAGE	The high order half word of the spool file ready date is not zero.
	CAUSE	The high order half word of the spool file ready date must be zero.
	ACTION	Initialize to zero the high-order half word of the spool file ready date.
-8053	MESSAGE	The value for *page to start* must be non-negative.
	CAUSE	The value for *page to start* cannot be negative.
	ACTION	Pass a valid *page to start* value.
-8054	MESSAGE	Cannot update the creator of an output spool file.
	CAUSE	Error occurred while trying to change the creator of an output spool file.
	ACTION	Contact Hewlett-Packard for support.
-8055	MESSAGE	Cannot change the state of an output spool file to the given state.
	CAUSE	It is illegal to change the state of the output spool file to the state given.
	ACTION	Not all spool file states are valid for the PUT operation.
-8056	MESSAGE	The only valid change of state for Spool file in CREATE state is to DEL_PENDING.
	CAUSE	It is illegal to change the spool file state from CREATE to other than DEL_PENDING.
	ACTION	Pass a valid state for the PUT.
-8057	MESSAGE	Cannot change a PRIVATE spool file to SPSAVE.
	CAUSE	Changing a PRIVATE spool file to SPSAVE is not allowed.
	ACTION	Don't try to SPSAVE a PRIVATE spool file.
-8058	MESSAGE	Cannot change the total number of copies to be printed for PRIVATE spool files.
	CAUSE	Changing the total number of copies of a PRIVATE spool file is not allowed.
	ACTION	Don't change the total number of copies of PRIVATE spool files.

-8059	MESSAGE	The *ready date* passed is invalid.
	CAUSE	The *ready date* passed is invalid.
	ACTION	Check the *ready date* passed.
-8060	MESSAGE	The *ready time* passed is invalid.
	CAUSE	The *ready time* passed is invalid.
	ACTION	Check the *ready time* passed.
-8061	MESSAGE	Cannot retrieve the capability of the caller from JMAT.
	CAUSE	Error occurred while getting the capability of the caller from JMAT.
	ACTION	Contact Hewlett-Packard for support.
-8062	MESSAGE	Cannot change the target device of a PRIVATE spool file.
	CAUSE	User does not have sufficient capability to change the target device of a PRIVATE spool file.
	ACTION	Get SM capability.
-8063	MESSAGE	The given target device is invalid.
	CAUSE	The given target device is invalid.
	ACTION	Check the target device passed.
-8064	MESSAGE	The target device is not output spoolable.
	CAUSE	The target device is not output spoolable.
	ACTION	Check the target device passed.
-8065	MESSAGE	The target device has not been openqed.
	CAUSE	The target device has not been openqed.
	ACTION	Check the target device.
-8066	MESSAGE	The given item number is not a valid GET item for input spool files.
	CAUSE	The item number is not valid for input spool files.
	ACTION	Don't use this item number on an input spool file.

-8067	MESSAGE	The given item number in the verification array is not valid for input spool files.
	CAUSE	The item number is not valid for input spool files.
	ACTION	Don't use this item number on an input spool file.
-8068	MESSAGE	The given item number is not a valid GET item for output spool files.
	CAUSE	The item number is not valid for output spool files.
	ACTION	Don't use this item number on an output spool file.
-8069	MESSAGE	The given item number in the verification array is not valid for output spool files.
	CAUSE	The item number is not valid for output spool files.
	ACTION	Don't use this item number on an output spool file.
-8070	MESSAGE	Cannot log the PUT operation on the spool file creator.
	CAUSE	Error occurred while calling transaction manager to log the PUT on spool file creator.
	ACTION	Contact Hewlett-Packard for support.
-8071	MESSAGE	Cannot update the spool file creator in the file label extension.
	CAUSE	Error occurred updating the spool file creator in the file label extension.
	ACTION	Contact Hewlett-Packard for support.
-8072	MESSAGE	Cannot update the spool file creator due to nesting call to the transaction manager aborted by AIF caller.
	CAUSE	PUT on the spool file creator failed because the nesting XM call aborted.
	ACTION	Contact Hewlett-Packard for support.
-8073	MESSAGE	Cannot complete IO to file label extension on disk. PUT might not have been completed.
	CAUSE	IO error occurred while writing the spool file information to FLABX.
	ACTION	Contact Hewlett-Packard for support.

-8074	MESSAGE	Failed to PUT the specified target device because spool file owner doesn't have access right to the device.
	CAUSE	Spool file owner doesn't have access to the device according to the ACD.
	ACTION	Alter the owner's access rights.
-8075	MESSAGE	Syntax error found in the given selection equation.
	CAUSE	Syntax error found in the given selection equation.
	ACTION	Correct the syntax error.
-8076	MESSAGE	Semantic error found in the given selection equation.
	CAUSE	Semantic error found in the given selection equation.
	ACTION	Correct the semantic error.
-8077	MESSAGE	Cannot optimize the selection equation.
	CAUSE	Error occurred while optimizing the selection.
	ACTION	Check the selection equation passed.
-8078	MESSAGE	Cannot retrieve the SPFDIR header record.
	CAUSE	Error occurred while getting the SPFDIR header record.
	ACTION	Contact Hewlett-Packard for support.
-8079	MESSAGE	Cannot find the SPFDIR entry.
	CAUSE	Error occurred while reading the SPFDIR.
	ACTION	Contact Hewlett-Packard for support.
-8080	MESSAGE	The given value for *defer* is out of range.
	CAUSE	The valid value for *defer* is either 0 or 1.
	ACTION	Pass a valid value for *defer*.
-8081	MESSAGE	The given spool file does not exist.
	CAUSE	The given spool file does not exist.
	ACTION	Check the file name passed.

-8082	MESSAGE	Error occurred while trying to copy and link the spool file.
	CAUSE	Error occurred while trying to copy and link the spool file.
	ACTION	Contact Hewlett-Packard for support.
-8083	MESSAGE	Invalid data address passed for *spf_ufid_array*.
	CAUSE	The address is not accessible to the caller.
	ACTION	Pass only address in accessible space.
-8084	MESSAGE	Invalid data address passed for *spf_id_array*.
	CAUSE	The address is not accessible to the caller.
	ACTION	Pass only address in accessible space.
-8085	MESSAGE	The give selection equation is longer than 277 characters.
	CAUSE	The CI parser cannot handle selection equation longer than 277 characters.
	ACTION	Don't pass selection equation longer than 277 characters.
-8086	MESSAGE	Error occurred while calling AIF system logging.
	CAUSE	Error occurred while calling AIF system logging.
	ACTION	Contact Hewlett-Packard for support.
-8087	MESSAGE	The address passed for overall status is not accessible to the caller. The calling application will be terminated with this error number.
	CAUSE	The address passed is not accessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-8088	MESSAGE	The address passed for the item status array is not accessible to the caller.
	CAUSE	The address passed is not accessible to the caller.
	ACTION	Pass only addresses in accessible spaces.
-8089	MESSAGE	The address passed for the verification item status array is not accessible to the caller.
	CAUSE	The address passed is not accessible to the caller.
	ACTION	Pass only addresses in accessible spaces.

-8090	MESSAGE	The address is not properly aligned for the data type to be accessed.
	CAUSE	The address is not properly aligned for the data type to be accessed.
	ACTION	Pass only variable that has the proper data alignment.

Spooler Process Information Warning Messages

+8500	MESSAGE	The device is not spoolable.
	CAUSE	The given device is not spoolable.
	ACTION	Check the device passed.
+8501	MESSAGE	Cannot openq on an inspooled device.
	CAUSE	Openq not allowed for inspool device.
	ACTION	Check the device passed.
+8502	MESSAGE	Cannot shutq on an inspooled device.
	CAUSE	Shutq not allowed for inspool device.
	ACTION	Check the device passed.
+8503	MESSAGE	The device is not spooled.
	CAUSE	The device is not spooled.
	ACTION	Check the device passed.
+8504	MESSAGE	The device is down.
	CAUSE	The device is down.
	ACTION	Check the device passed.
+8505	MESSAGE	The device is not available.
	CAUSE	The device is not available.
	ACTION	Check the device passed.
+8506	MESSAGE	The device has already been spooled.
	CAUSE	The device has already been spooled.
	ACTION	Check the device passed.
+8507	MESSAGE	Cannot reserve the device.
	CAUSE	Error occurred when trying to reserve the device for NMS.
	ACTION	Check the device passed.

+8508	MESSAGE	Cannot stopspool because of the current spooler process state.
	CAUSE	It is illegal to stopspool on the current state of the spooler.
	ACTION	Check the device passed.
+8509	MESSAGE	Cannot suspendspool because of the current spooler process state.
	CAUSE	It is illegal to suspendspool on the current state of the spooler.
	ACTION	Check the device passed.
+8510	MESSAGE	Cannot resumespool because of the current spooler process state.
	CAUSE	It is illegal to resumespool on the current state of the spooler.
	ACTION	Check the device passed.
+8511	MESSAGE	Cannot releasespool because of the current spooler process state.
	CAUSE	It is illegal to releasespool on the current state of the spooler.
	ACTION	Check the device passed.
+8512	MESSAGE	Cannot suspendspool on an inspool device.
	CAUSE	Suspendspool is not allowed on an inspool device.
	ACTION	Check the device passed.
+8513	MESSAGE	Cannot resumespool on an inspool device.
	CAUSE	Resumespool is not allowed on an inspool device.
	ACTION	Check the device passed.
+8514	MESSAGE	Cannot releasespool on an inspool device.
	CAUSE	Releasespool is not allowed on an inspool device.
	ACTION	Check the device passed.
+8515	MESSAGE	Cannot resumespool on the device if *nokeep* has been specified.
	CAUSE	Resumespool is not allowed if *nokeep* has been specified on stopspool.
	ACTION	Don't resumespool on a nokeep device.

+8516	MESSAGE	*Offset* cannot be specified with this particular device.
	CAUSE	*Offset* cannot be specified with this particular device.
	ACTION	The offset is ignored.
+8517	MESSAGE	Error occurred while altering the list device in JMAT.
	CAUSE	Error occurred while altering the list device in JMAT.
	ACTION	Contact Hewlett-Packard for support.
+8518	MESSAGE	More qualifying entry found by AIFSPFLIST than can be returned in the array.
	CAUSE	The given array is too small.
	ACTION	Pass a larger array.
+8519	MESSAGE	Mismatching spool file ID and address.
	CAUSE	The spool file ID and address do not identify the same spool file. A spool file may have been purged and the address reused by a newly created spool file.
	ACTION	Check the spool file ID passed.
+8520	MESSAGE	No active files were suspended.
	CAUSE	No active spool files when SUSPEND was executed.
	ACTION	No action needed.
+8521	MESSAGE	Undefined warning messages returned from suspending the spooler.
	CAUSE	Unknown.
	ACTION	Check the device passed.
+8522	MESSAGE	*spf_count* was zero and *stop_search* was true, nothing was performed.
	CAUSE	*spf_count* equal zero and *stop_search* was set to true.
	ACTION	Modify either *spf_count* of *stop_search* flag, depending on the need.

KSAM Status Messages

-9001	MESSAGE	An internal error was encountered while moving data to/from the user buffer.
	CAUSE	Same as the message content.
	ACTION	Check the buffer parameter and its bounds, or contact your Hewlett-Packard support representative.
-9002	MESSAGE	Unexpected error encountered while prefetching file pages.
	CAUSE	Could be out of disc space, out of file limit or any other unexpected internal error.
	ACTION	Check for free disk space, or call your Hewlett-Packard support representative.
-9003	MESSAGE	The EOF status was encountered while accessing the file.
	CAUSE	The end-of-file was reached while accessing a file.
	ACTION	The file is ready to be closed. Call FCLOSE to close the file.
-9004	MESSAGE	The size of the user buffer is too small to hold all the required data.
	CAUSE	The buffer used by `AIFKSMCREATE` of the first `AIFKSMREAD` is too small to hold the required information.
	ACTION	Increase the buffer size based on the KSAM AIF documentation.
-9005	MESSAGE	The access to the KSAM file violates the access rights specified in the file opening.
	CAUSE	The access options specified in the file opening does not allow this access.
	ACTION	Refer to the KSAM AIF documentation for using the correct access options in the file opening.
-9006	MESSAGE	The user does not have the priviledge capability to call this AIF intrinsic.
	CAUSE	Same as the message content.
	ACTION	Obtain the priviledge capability for the user's program.
-9007	MESSAGE	Unexpected status from HPFOPEN was received while creating the KSAM file structure.
	CAUSE	Same as the message content.
	ACTION	Contact you Hewlett-Packard support representative.

-9008	MESSAGE	Volumeset specified in the *vol_set_name* parameter does not exist on the system.
	CAUSE	Same as the message content.
	ACTION	Specify a valid volumeset.
-9009	MESSAGE	Volume specified in the *vol_name* parameter does not exist on the specified volumeset. MPEXL_SYSTEM_VOLUME_SET is the default volume_set if none was specified.
	CAUSE	Same as the message content.
	ACTION	Specify another volume.
-9010	MESSAGE	Volume class specified in the *vol_class* parameter does not exist on the specified. MPEXL_SYSTEM_VOLUME_SET is the default volume_set if none was specified.
	CAUSE	Same as the message content.
	ACTION	Specify a valid volume class.
-9011	MESSAGE	Specified LDEV is not mounted at this time.
	CAUSE	You specified a device that is not mounted.
	ACTION	Specify a mounted device.
-9012	MESSAGE	Volume specified in the *vol_name* parameter is in the progress of mounting.
	CAUSE	The volume is being mounted now.
	ACTION	Wait after the volume mounted.
-9013	MESSAGE	Unexpected status from Volume Management was received. This was probably due to having specified *vol_name*, *vol_class*, *ldev_num*, or *vol_set_name* parameters with conflicting names.
	CAUSE	An internal volume management error occurred due to name conflicts.
	ACTION	Check the device parameters and retry.
-9014	MESSAGE	The volume class specified in the *vol_class* parameter does not have any member volumes mounted on the system. For more information, use the DSTAT command.
	CAUSE	You specified an empty volume class.
	ACTION	Specify a volume class with a mounted volume.

-9015	MESSAGE	Internal Error. Pathname in the buffer passed to AIFKSMCREATE is bad.
	CAUSE	The pathname in the buffer passed to AIFKSMCREATE is bad.
	ACTION	Check the application to make sure you are passing the correct buffer.
-9016	MESSAGE	The directory parameter does not have a valid pathname syntax.
	CAUSE	The syntax of the directory parameter is not a valid pathname.
	ACTION	Check the application and pass in a valid directory parameter.
-9017	MESSAGE	The HFS filename is > 16 characters; cannot be created in a MPE group/account.
	CAUSE	The HFS filename is not a valid MPE name.
	ACTION	Create the file in a HFS directory.
-9018	MESSAGE	Must specify both group/account when creating a HFS file in an MPE directory.
	CAUSE	Creating a HFS file in a MPE directory requires both the group/account parms.
	ACTION	Change the application.
-9019	MESSAGE	File must be a KSAM/XL File.
	CAUSE	This AIF is only valid for KSAM/XL files.
	ACTION	Check the application to make sure you are passing in the correct filenumber.
-9020	MESSAGE	The read probe failed on the directory parameter.
	CAUSE	You do not have the access rights to access the directory address and length.
	ACTION	Make sure you are not passing in a bad length in the first word of directory.
-9021	MESSAGE	File number is bad.
	CAUSE	The file number passed in does not exist.
	ACTION	Check the application to see if the FOPEN was successful.

-9022	MESSAGE	Internal error while reading a file label extension.
	CAUSE	Internal error while reading a file label extension.
	ACTION	Contact your Hewlett-Packard Support Representative.
-9023	MESSAGE	Internal error while writing a file label extension.
	CAUSE	Internal error while writing a file label extension.
	ACTION	Contact your Hewlett-Packard Support Representative.
-9024	MESSAGE	Internal error while getting the length of a file label extension.
	CAUSE	Internal error while getting the length of a file label extension.
	ACTION	Contact your Hewlett-Packard Support Representative.
-9025	MESSAGE	Internal error while creating a file label extension.
	CAUSE	Internal error while creating a file label extension.
	ACTION	Contact your Hewlett-Packard Support Representative.

System wide Information Status Messages

-10001	MESSAGE	Invalid item number passed in *itemnum_array* for AIFSYSWIDEGET.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate item number as specified in the reference manual.
-10003	MESSAGE	An account was not specified for AIFSYSWIDEGET.
	CAUSE	The account name must be specified to obtain account information.
	ACTION	An *item_array* element associated with the account item number is needed.
-10004	MESSAGE	User and group information cannot be obtained in the same AIFSYSWIDEGET call.
	CAUSE	The *itemnum_array* contained items for user and group.
	ACTION	If group and user information is needed use separate AIFSYSWIDEGET calls.
-10005	MESSAGE	Invalid output priority in AIFSYSWIDEGET call.
	CAUSE	The output priority value in the *item_array* is invalid.
	ACTION	Use a value from 0 to 14 in *item_array*.
-10006	MESSAGE	Invalid spool file state in AIFSYSWIDEGET call.
	CAUSE	The spool file state specified in *item_array* was invalid.
	ACTION	Use a value from 0,1,2,3,4,6,7,8 in *item_array*.
-10007	MESSAGE	Invalid spool file inout value in AIFSYSWIDEGET call.
	CAUSE	The spool file inout value specified in *item_array* was invalid.
	ACTION	Use a value from 0 or 1 in *item_array*.
-10008	MESSAGE	Invalid spool file pages value in AIFSYSWIDEGET call.
	CAUSE	The spool file pages value specified in *item_array* was invalid.
	ACTION	Use a value from 0 to 65535 in *item_array*.

-10009	MESSAGE	Invalid spool file copies value in AIFSYSWIDEGET call.
	CAUSE	The spool file copies value specified in *item_array* was invalid.
	ACTION	Use a value from 0 to 65535 in *item_array*.
-10010	MESSAGE	Invalid spool file jobabort value in AIFSYSWIDEGET call.
	CAUSE	The spool file jobabort value specified in *item_array* was invalid.
	ACTION	Use a value 0 or 1 in *item_array*.
-10011	MESSAGE	Invalid spool file disposition in AIFSYSWIDEGET call.
	CAUSE	The spool file disposition specified in *item_array* was invalid.
	ACTION	Use a value 1 or 2 in *item_array*.
-10012	MESSAGE	Invalid search key.
	CAUSE	The search key pointer is inaccessible for write access.
	ACTION	Check search key to make sure it is in an accessible space.
-10014	MESSAGE	The *num_entries* specified is too large.
	CAUSE	The *num_entries* specified is too large for the return arrays.
	ACTION	Specify a smaller *num_entries* parameter.
-10015	MESSAGE	Badly aligned buffer_ptr was encountered.
	CAUSE	The buffer_ptr passed to AIFSYSWIDEGET is unaligned.
	ACTION	Check the application for alignment of buffer_ptr.
-10016	MESSAGE	Num_entries is a required parameter.
	CAUSE	Num_entries is a required parameter to AIFSYSWIDEGET.
	ACTION	Check the application.
-10017	MESSAGE	Search key is not valid with the pathname specified.
	CAUSE	The search key pathname is not a member of the fileset specified.
	ACTION	Check the application.
-10018	MESSAGE	Num_entries should not be zero when return arrays are passed.
	CAUSE	Non-nil return array parameters were passed.
	ACTION	Check the logic of your application.

Ports Management Status Messages

-11001	MESSAGE	Invalid itemnum specified in itemnums array.
	CAUSE	Item number is not valid for this call.
	ACTION	Check the logic of your application.
-11002	MESSAGE	Invalid create option.
	CAUSE	The value given for the create option is not a valid value.
	ACTION	Check the logic of your application.
-11003	MESSAGE	The Max message size parm is more than the maximum allowed, or less than 0.
	CAUSE	The value for *max_msg_size* is out of range.
	ACTION	Check the logic of your application.
-11004	MESSAGE	Normal message size is more than the maximum allowed, or less then zero.
	CAUSE	The value for *normal_msg_size* is out of range.
	ACTION	Check the logic of your application.
-11005	MESSAGE	Max number of messages is less than 0 or more than the maximum allowed.
	CAUSE	The *max_norm_msgs* value is out of range.
	ACTION	Check the logic of your application.
-11006	MESSAGE	Normal message size specified is more than the maximum message size.
	CAUSE	Normal size cannot be larger than maximum size.
	ACTION	Check the logic of your application.
-11007	MESSAGE	Num normal msgs * norm_msg_size NOT >= max_msg_size.
	CAUSE	Must have room for at least one max sized message.
	ACTION	Check the logic of your application.

-11008	MESSAGE	Timeout value not >= -1.
	CAUSE	Timeout value is out of range.
	ACTION	Check the logic of your application.
-11009	MESSAGE	Priority value not between 0 and 31 for AIFPORTSEND.
	CAUSE	Priority is out of range.
	ACTION	Check the logic of your application.
-11010	MESSAGE	Open of existing port failed; port does not exist.
	CAUSE	Opened old port, but port does not exist.
	ACTION	Check the logic of your application.
-11011	MESSAGE	Open of new port failed; port already exists.
	CAUSE	Opened new port, but port already exists.
	ACTION	Check the logic of your application.
-11012	MESSAGE	Max msg size specified > max size when port was created.
	CAUSE	Port was created with a smaller max size than passed in this call.
	ACTION	Check the logic of your application.
-11013	MESSAGE	*Num_norm_msgs* specified is more than originally allowed.
	CAUSE	*Num_norm_msgs* is greater than number specified when port was created.
	ACTION	Check the logic of your application.
-11014	MESSAGE	Invalid creation option for an asynchronous port.
	CAUSE	Item 11201 was not specified with a value of 2.
	ACTION	Check the logic of your application.
-11015	MESSAGE	Invalid option, asynchronous ports cannot be permanent.
	CAUSE	Item 11205 cannot be specified with an async port.
	ACTION	Check the logic of your application.

-11016	MESSAGE	Invalid option combination between connectionless/waited.
	CAUSE	Connectionless sends must specify item 11101 with a value of -1 nowait.
	ACTION	Check the logic of your application.
-11017	MESSAGE	Invalid port was specified to AIFPORTINT.
	CAUSE	An invalid asynchronous port id was specified.
	ACTION	Check the logic of your application.
-11018	MESSAGE	Invalid creation option for asynchronous port.
	CAUSE	Item 11207 interrupt state was specified without item 11206 handler address.
	ACTION	Check the logic of your application.
-11019	MESSAGE	Invalid send to an asynchronous port, no receiver exists.
	CAUSE	The receiver has closed the asynchronous port before the sender, or the receiver process has terminated.
	ACTION	Close the port and check the logic of your application.
-11020	MESSAGE	A send was issued against a port which no longer exists.
	CAUSE	A portid was specified for a port which is no longer open.
	ACTION	Check the logic of your application.
-11023	MESSAGE	Invalid option combination portid = 0 and item 11007 is true.
	CAUSE	A single asynchronous portid is required when item 11007 is used on the AIFPORTRECEIVE call.
	ACTION	Check the logic of your application.
-11024	MESSAGE	An invalid receive option was used on a synchronous port.
	CAUSE	Item 11007 can only be used an asynchronous portid.
	ACTION	Check the logic of your application.
-11025	MESSAGE	There are no more messages with pending interrupts. No message was received.
	CAUSE	There currrently are no messages with pending interrups on the port.
	ACTION	Check the logic of your application.

Device Status Messages

-13001	MESSAGE	No Ldev or Device-key specified for Device Get.
	CAUSE	Neither an ldev or device-key was specified.
	ACTION	Call **AIFSYSWIDEGET** to obtain a ldev or device-key.
-13002	MESSAGE	Invalid Device Get item number specified.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate item number.
-13003	MESSAGE	Invalid Ldev number specified.
	CAUSE	The ldev specified is invalid or not configured.
	ACTION	Call **AIFSYSWIDEGET** to obtain a valid ldev.
-13004	MESSAGE	Invalid device-key specified.
	CAUSE	The device-key specified is invalid or not configured.
	ACTION	Call **AIFSYSWIDEGET** to obtain a valid device-key.
-13005	MESSAGE	The item number specified is incompatible with the LDEV.
	CAUSE	The item number is not compatible with the device type.
	ACTION	Use the item number which is compatible with the device type.
-13006	MESSAGE	Internal Error.
	CAUSE	Can not access the Device Class Table.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-13007	MESSAGE	Fail to convert UFID to logical device number.
	CAUSE	Can not open the device file with the given UFID.
	ACTION	Use a valid UFID as input value.

-13008	MESSAGE	Internal Error.
	CAUSE	Can not close/purge a disc file.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-13009	MESSAGE	Fail to perform a control operation to a terminal device file.
	CAUSE	The terminal may be in an inappropriate state or the operation is not supported.
	ACTION	Contact Hewlett-Packard for support.
-13010	MESSAGE	Fail to perform a control operation to a printer device file.
	CAUSE	The printer may be in an inappropriate state or the operation is not supported.
	ACTION	Check the device state or the functionality requested.
-13011	MESSAGE	Fail to perform a control operation to a tape device file.
	CAUSE	The tape may be in an inappropriate state or the operation is not supported.
	ACTION	Contact Hewlett-Packard for support.
-13012	MESSAGE	Fail to perform a control operation to a disc device file.
	CAUSE	The disc may be in an inappropriate state or the operation is not supported.
	ACTION	Check the device state or the functionality requested.
-13013	MESSAGE	The device type is incompatible with the requested functionality.
	CAUSE	The control operation can not be performed on this type of device.
	ACTION	Do not request the control operation on this device type.
-13014	MESSAGE	The device is not configured.
	CAUSE	The device is not configured.
	ACTION	Check the IO configuration.

-13015	MESSAGE	The given item number is not a valid PUT item for device PUT.
	CAUSE	The item number is not valid for device PUT.
	ACTION	Pass an appropriate item number.
-13016	MESSAGE	The given item number in the verification array is not valid for AIFDEVICEPUT
	CAUSE	An invalid item number was passed in the verification array.
	ACTION	Pass an appropriate item number.
-13017	MESSAGE	The given item value in the items_array is not valid for AIFDEVICEGET/PUT.
	CAUSE	An invalid item value was passed.
	ACTION	Pass an appropriate item value.
-13019	MESSAGE	Failed to perform the requested device control operation.
	CAUSE	AIF can not complete the request because of IO problems.
	ACTION	Check to see if the requested operation is valid for the device type.
-13020	MESSAGE	The device is not available.
	CAUSE	For printer, serial printer, or tape device, the device should have already been opened. For terminal device, a problem occured when AIF tried to open the device.
	ACTION	Check the availability of the device.
-13022	MESSAGE	Failed to access the device.
	CAUSE	An error was returned while trying to access the device.
	ACTION	Verify that the device number is correct.
-13023	MESSAGE	Internal error.
	CAUSE	JSINFO returned a bad status.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.

-13024	MESSAGE	Unable to return security logon attempt information.
	CAUSE	The security information was not found.
	ACTION	The HP 3000 Security Monitor/iX is not found. Check with the system manager.
-13025	MESSAGE	Unable to return security terminal password information.
	CAUSE	The security information was not found.
	ACTION	The HP 3000 Security Monitor/iX is not found. Check with the system manager.
-13501	MESSAGE	No Device Class Name or Device Class Key specified for Device Class Get.
	CAUSE	Neither a device class name or key was specified.
	ACTION	Call AIFSYSWIDEGET to obtain a device class name or key.
-13502	MESSAGE	Invalid Device Class Get item number specified.
	CAUSE	A non-zero, invalid item number was passed in *itemnum_array*.
	ACTION	Pass an appropriate item number and end with a zero.
-13503	MESSAGE	Internal Error.
	CAUSE	Unexpected ESCAPE from table locking in AIFDEVCLASSGET.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-13504	MESSAGE	Internal Error.
	CAUSE	Unexpected ESCAPE from table unlocking in AIFDEVCLASSGET.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.
-13505	MESSAGE	Mismatching device key and name.
	CAUSE	The device key and the device name passed do not identify the same device.
	ACTION	Check the device key and the device name.
-13506	MESSAGE	The I/O device class returned is not recognized by AIF.
	CAUSE	AIF internal error.
	ACTION	Contact your Hewlett-Packard support representative and be prepared to provide information on how to reproduce the problem.

-17001	MESSAGE	Media manager process not created.
	CAUSE	The program HPOPTMGR.PUB.SYS program was not run at system startup.
	ACTION	Run HPOPTMGR.PUB.SYS and retry AIF operation.
-17002	MESSAGE	Unable to open the media information file on optical disk.
	CAUSE	The media information file on the optical disk could not be opened.
	ACTION	Re-initialize media via the MOUTIL INITMO command and retry AIF operation.
-17003	MESSAGE	MOUTIL SYNCTABLE in progress by another process. Try again later.
	CAUSE	A SYNCTABLE was invoked by another process using the optical disk library system.
	ACTION	Retry AIF operation later.
-17004	MESSAGE	Invalid PIN specified.
	CAUSE	AIF specified an invalid PIN for operation.
	ACTION	Retry AIF operation with valid PIN - PIN of process who has drive allocated.
-17005	MESSAGE	Invalid optical disk drive ldev specified.
	CAUSE	AIF specified an invalid optical disk drive ldev.
	ACTION	Retry AIF operation with valid optical disk drive ldev.
-17006	MESSAGE	Optical disk ldev currently in use by another process.
	CAUSE	The optical disk ldev is currently in use by another process.
	ACTION	Retry the AIF operation or use another optical disk ldev.
-17007	MESSAGE	Optical disk ldev currently not allocated.
	CAUSE	The optical disk ldev must be allocated for this AIF operation.
	ACTION	Allocate the optical disk ldev and retry the AIF operation.

-17008	MESSAGE	All drives are currently allocated.
	CAUSE	An allocate by media name was performed and all drives accessible to the media are currently allocated.
	ACTION	Retry the AIF operation when a drive is available.
-17009	MESSAGE	Media currently mounted in the specified drive.
	CAUSE	A mount was performed on a drive that already had media mounted.
	ACTION	Wait for media to be dismounted before attempting mount.
-17010	MESSAGE	Expected media not mounted in expected optical disk ldev.
	CAUSE	Expected media not mounted in expected optical disk ldev.
	ACTION	Mount media in the optical disk ldev and retry the AIF operation. If this fails, perform the MOUTIL SYNCTABLE command for this optical disk ldev.
-17011	MESSAGE	Requested optical disk library media currently in use.
	CAUSE	The requested media is currently being used.
	ACTION	Retry the AIF operation with another piece of media.
-17012	MESSAGE	Requested optical media not found.
	CAUSE	The requested optical media can not be found within the optical library system.
	ACTION	Retry the AIF operation specifying a different piece of media.
-17013	MESSAGE	Requested optical media currently in use.
	CAUSE	The requested optical media is currently being used.
	ACTION	Retry the AIF operation specifying a different piece of media.
-17014	MESSAGE	Media label found, but no drive currently available to allocate.
	CAUSE	An allocate by media name was performed and all drives accessible to the media are currently allocated.
	ACTION	Retry the AIF operation when a drive is available.

-17015	MESSAGE	Verification of media label failed.
	CAUSE	Specificied media label does not match media label for the mounted media.
	ACTION	Retry AIF operation with a different media label to verify against.
-17016	MESSAGE	Volume close error occurred.
	CAUSE	A volume close error occurred during dismount of volume set. Files possibly still opened on the volume set.
	ACTION	Once all files are closed on the volume set invoke AIFDISMOMOUNT to dismount the media.
-17017	MESSAGE	Media Manager class error occurred.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17018	MESSAGE	Autochanger error occurred.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17019	MESSAGE	Magneto-optical drive error occurred.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17020	MESSAGE	Magneto-optical drive allocation error occurred.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17021	MESSAGE	General device error.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.

-17022	MESSAGE	Media moving error.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17023	MESSAGE	Media label operation error.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17024	MESSAGE	Volume Management error.
	CAUSE	Internal error.
	ACTION	Contact your Hewlett Packard support representative.
-17025	MESSAGE	Media Manager internal error.
	CAUSE	Media manager returned an error, but from a another Subsys.
	ACTION	Contact your Hewlett Packard support representative.
-17026	MESSAGE	Invalid item number passed in itemnum_array to AIFMOALLOCATE.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-17027	MESSAGE	Invalid item number passed in itemnum_array to AIFMODEALLOCATE.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-17028	MESSAGE	Invalid item number passed in itemnum_array to AIFMOMOUNT.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.

-17029	MESSAGE	Invalid item number passed in itemnum_array to AIFMODISMOUNT.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-17030	MESSAGE	Unable to get the pointer to magneto-optical AIF known process object.
	CAUSE	GET_KPO_POINTER returned a bad status.
	ACTION	Contact your Hewlett Packard support representative.
-17031	MESSAGE	Unable to create the magneto-optical AIF known process object.
	CAUSE	CREATE_OBJECT returned a bad status.
	ACTION	Contact your Hewlett Packard support representative.
-17032	MESSAGE	Unable to create the media manager reply port.
	CAUSE	CREATE_PORT returned a bad status.
	ACTION	Contact your Hewlett Packard support representative.
-17033	MESSAGE	Unable to get the media manager port number.
	CAUSE	The optical media manager is not running due to not having an optical disk library system configured, the media manager encountering a fatal error, or the program HPOPTMGR.PUB.SYS not being run.
	ACTION	Verify that a optical disk library system is configured, investigate any media manager errors displayed on the console, run the program HPOPTMGR.PUB.SYS.
-17034	MESSAGE	Unable to set the pointer to magneto-optical AIF known process object.
	CAUSE	SET_KPO_POINTER returned a bad status.
	ACTION	Contact your Hewlett Packard support representative.
-17035	MESSAGE	The maximum number of nowait requests have been made for this pin.
	CAUSE	The maximum number of combined nowait mounts or dismounts that can be outstanding for a pin is 32.
	ACTION	Complete outstanding nowait mounts or dismounts and retry AIF operation.

-17036	MESSAGE	Invalid nowait identifier specified.
	CAUSE	An invalid nowait identifier was specified.
	ACTION	Specify a valid nowait identifier and retry AIF operation.
-17037	MESSAGE	Only one of optical drive or media label can be specified.
	CAUSE	Both optical drive and media label were specified.
	ACTION	Specify only one of optical drive or media label and retry AIF operation.
-17038	MESSAGE	Invalid value for prompt for media item.
	CAUSE	Prompt for media value must be 1,2, or 3.
	ACTION	Specify valid prompt for media item and retry AIF operation.
-17039	MESSAGE	Volume set item invalid when nowait item specified with initialization value.
	CAUSE	Volume set item can not be specified when nowait item is specified with initialization value.
	ACTION	Specify only one of volume set item or nowait item with initialization value and retry AIF operation.
-17040	MESSAGE	Error opening the media manager database.
	CAUSE	The media manager database could not be opened. Possibly due to the optical media manager not running. This could be due to not having an optical disk library system configured, the media manager encountering a fatal error, or the program HPOPTMGR.PUB.SYS not being run.
	ACTION	Verify that a optical disk library system is configured, investigate any media manager errors displayed on the console, run the program HPOPTMGR.PUB.SYS.
-17041	MESSAGE	Invalid autochanger or mounted optical disk ldev specified.
	CAUSE	An invalid autochanger or mounted optical disk ldev was specified. The optical media manager may not be running.
	ACTION	Specify a valid autochanger or mounted optical disk ldev and retry AIF operation.
-17042	MESSAGE	Autochanger not configured.
	CAUSE	The specified autochanger is not configured on the system. The optical media manager may not be running.
	ACTION	Retry AIF operation with a configured autochanger.

-17043	MESSAGE	MOUTIL SYNCTABLE required or currently in progress by another process.
	CAUSE	A SYNCTABLE was invoked by another process using the optical disk library system or a SYNCTABLE is required.
	ACTION	Perform MOUTIL SYNCTABLE and retry operation later.
-17044	MESSAGE	Optical drive not allocated by specified PIN.
	CAUSE	AIF specified an invalid PIN for operation.
	ACTION	Retry AIF operation with valid PIN - PIN of process who has drive allocated.
-17045	MESSAGE	Media not mounted on optical drive ldev.
	CAUSE	Media not mounted on optical drive ldev.
	ACTION	Mount media in the optical disk ldev and retry the AIF operation.
-17046	MESSAGE	Error obtaining the mounted volume entry for the specified ldev.
	CAUSE	VLM_OBTAIN_MVT_ENTRY returned a bad status.
	ACTION	Contact your Hewlett Packard support representative.
-17047	MESSAGE	Mounted media unavailable.
	CAUSE	VLM_OBTAIN_MVT_ENTRY returned status that the state of the mounted volume is unavailable.
	ACTION	Contact your Hewlett Packard support representative.
-17048	MESSAGE	Unable to open the media information file on optical disk.
	CAUSE	The media information file on the optical disk could not be opened.
	ACTION	Re-initialize media via the MOUTIL INITMO command and retry AIF operation.
-17049	MESSAGE	Invalid optical disk drive ldev specified.
	CAUSE	AIF specified an invalid optical disk drive ldev.
	ACTION	Retry AIF operation with valid optical disk drive ldev.

-17050	MESSAGE	Invalid autochanger specified.
	CAUSE	AIF specified an invalid autochanger.
	ACTION	Retry AIF operation with valid autochanger.
-17051	MESSAGE	Invalid lower range for storage slot info specified. Lower range must be greater than or equal to one and less than or equal to the upper range.
	CAUSE	Invalid lower range for storage slot info specified.
	ACTION	Retry AIF operation with valid lower range for storage slot info.
-17052	MESSAGE	Invalid upper range for storage slot info specified.
	CAUSE	Invalid upper range for storage slot info specified. Upper range must be greater than or equal to one.
	ACTION	Retry AIF operation with valid upper range for storage slot info.
-17053	MESSAGE	Invalid item number passed in itemnum_array to AIFMOGET.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-17054	MESSAGE	Invalid item number passed in ver_itemnum_array to AIFMOPUT.
	CAUSE	A non-zero invalid item number was passed in ver_itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-17055	MESSAGE	Invalid item number passed in itemnum_array to AIFMOPUT.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with a zero.
-17056	MESSAGE	Invalid media label specified.
	CAUSE	The first character of the media name, subname1, or subname2, is "@".
	ACTION	Pass a media name, subname1, or subname2 whose first character is not "@".

+17501	MESSAGE	The specified optical drive is not allocated.
	CAUSE	This warning will be returned when AIFMODEALLOCATE is called to deallocate a drive that has not been previously allocated.
	ACTION	No action.
+17502	MESSAGE	The specified optical drive already hard allocated by PIN.
	CAUSE	This warning will be returned when AIFMOALLOCATE is called to allocate a drive that has already been allocated by the requesting PIN.
	ACTION	No action.
+17503	MESSAGE	Magneto-Optical drive allocation warning occurred.
	CAUSE	Internal Warning.
	ACTION	No action.
+17504	MESSAGE	Autochanger unlock warning occurred.
	CAUSE	Internal Warning.
	ACTION	No action.
+17505	MESSAGE	Autochanger lock warning occurred.
	CAUSE	Internal Warning.
	ACTION	No action.
+17506	MESSAGE	Pin item has already been specified.
	CAUSE	Pin item was specified more than once in the AIF request. The first pin item specified will be used.
	ACTION	No action.
+17507	MESSAGE	Ldev item has already been specified.
	CAUSE	Ldev item was specified more than once in the AIF request. The first ldev item specified will be used.
	ACTION	No action.

+17508	MESSAGE	Media label item has already been specified.
	CAUSE	Media label item was specified more than once in the AIF request. The first media label item specified will be used.
	ACTION	No action.
+17509	MESSAGE	Prompt item has already been specified.
	CAUSE	Prompt item was specified more than once in the AIF request. The first prompt item specified will be used.
	ACTION	No action.
+17510	MESSAGE	Volume set item has already been specified.
	CAUSE	Volume item was specified more than once in the AIF request. The first volume item specified will be used.
	ACTION	No action.
+17511	MESSAGE	Nowait item has already been specified.
	CAUSE	Nowait item was specified more than once in the AIF request. The first nowait item specified will be used.
	ACTION	No action.
+17512	MESSAGE	MOUTIL SYNCTABLE in progress by another process. Data may be invalid.
	CAUSE	A SYNCTABLE was invoked by another process using the optical disk library system.
	ACTION	Retry AIF operation later.
+17513	MESSAGE	Media label verify item has already been specified.
	CAUSE	Media label verify item was specified more than once in the AIF request. The first media label verify item specified will be used.
	ACTION	No action.
+17514	MESSAGE	Volume set verify item has already been specified.
	CAUSE	Volume set verify item was specified more than once in the AIF request. The first volume set verify item specified will be used.
	ACTION	No action.

Status Messages for Workgroup Information

-19001	MESSAGE	A non-existent workgroup cannot be purged.
	CAUSE	An attempt was made to purge a non-existent workgroup.
	ACTION	Check the workgroup name.
-19002	MESSAGE	A system default workgroup cannot be purged.
	CAUSE	An attempt was made to purge a system default workgroup.
	ACTION	Check the workgroup name.
-19003	MESSAGE	A non-existent workgroup cannot be modified.
	CAUSE	The workgroup name passed does not exist.
	ACTION	Check the workgroup name.
-19004	MESSAGE	Number of member processes could not be returned from non-existent workgroup.
	CAUSE	The workgroup name passed does not exist.
	ACTION	Check the workgroup name.
-19005	MESSAGE	A list of member processes could not be returned from non-existent workgroup.
	CAUSE	The workgroup name passed does not exist.
	ACTION	Check the workgroup name.
-19006	MESSAGE	Cannot get quantum from a non-existent workgroup.
	CAUSE	The workgroup name passed does not exist.
	ACTION	Check the workgroup name.
-19007	MESSAGE	Cannot get workgroup information for a non-existent pin.
	CAUSE	The pin passed does not exist.
	ACTION	Check the pin.
-19008	MESSAGE	Cannot assign a process to a non-existent workgroup.
	CAUSE	The workgroup name passed does not exist.
	ACTION	Check the workgroup name.

-19009	MESSAGE	Base priority too high.
	CAUSE	Base priority should be in the range of 127 to 13567.
	ACTION	Pass a base priority within the range.
-19010	MESSAGE	Base priority too low.
	CAUSE	Base priority should be in the range of 127 to 13567
	ACTION	Pass a base priority within the range.
-19011	MESSAGE	Invalid Limit priority value.
	CAUSE	Limit priority higher than base priority.
	ACTION	Pass a correct limit priority.
-19012	MESSAGE	Limit priority too low.
	CAUSE	Limit priority should be in the range of 127 to 13567.
	ACTION	Pass an appropriate value within the correct range for Limit Priority.
-19013	MESSAGE	Minimum quantum value higher than valid value range.
	CAUSE	Minimum Quantum should be in the range of 1..32767.
	ACTION	Pass an appropriate value within the correct range for Minimum Quantum value.
-19014	MESSAGE	Minimum quantum value lower than valid value range.
	CAUSE	Minimum Quantum should be in the range of 1..32767.
	ACTION	Pass an appropriate value within the correct range for Minimum Quantum value.
-19015	MESSAGE	Maximum quantum value higher than valid value range.
	CAUSE	Maximum Quantum should be in the range of 1..32767.
	ACTION	Pass an appropriate value within the correct range for Minimum Quantum value.
-19016	MESSAGE	Maximum quantum value lower than valid value range.
	CAUSE	Maximum Quantum should be in the range of 1..32767.
	ACTION	Pass an appropriate value within the correct range for Minimum Quantum value.

-19017	MESSAGE	Invalid Boost Property value.
	CAUSE	Boost property should be 0 for DECAY or 1 for OSCILLATE.
	ACTION	Pass an appropriate value within the correct range for Boost Property value.
-19018	MESSAGE	Invalid Timeslice type.
	CAUSE	Timeslice is an integer type.
	ACTION	Pass an appropriate number for Timeslice value.
-19019	MESSAGE	Invalid Timeslice value.
	CAUSE	Timeslice is a multiple of 100 milliseconds and has a minimum value of 100 milliseconds.
	ACTION	Pass an appropriate value within the correct range for Timeslice value.
-19020	MESSAGE	Timeslice value too high.
	CAUSE	Timeslice is a multiple of 100 milliseconds and has a maximum value of 32700 milliseconds.
	ACTION	Pass an appropriate value within the correct range for Timeslice value.
-19021	MESSAGE	Timeslice value too low.
	CAUSE	Timeslice is a multiple of 100 milliseconds and has a minimum value of 100 milliseconds.
	ACTION	Pass an appropriate value within the correct range for Timeslice value.
-19022	MESSAGE	Invalid workgroup name for position parameter in AIFWGADD.
	CAUSE	The workgroup does not exist.
	ACTION	Pass an existing workgroup name.
-19023	MESSAGE	Minimum CPU Percentage value less than zero.
	CAUSE	The value can range from 0% to 100%.
	ACTION	Pass an appropriate value within the correct range for Minimum CPU Percentage value.
-19024	MESSAGE	Minimum CPU Percentage value greater than 100.
	CAUSE	The value can range from 0% to 100%.
	ACTION	Pass an appropriate value within the correct range for Minimum CPU Percentage value.

-19025	MESSAGE	Invalid Minimum CPU Percentage value.
	CAUSE	The value causes total percentage to be more than 100.
	ACTION	Pass an appropriate value for Minimum CPU Percentage value.
-19026	MESSAGE	Maximum CPU Percentage value less than zero.
	CAUSE	The value can range from 0% to 100%.
	ACTION	Pass an appropriate value within the correct range for Maximum CPU Percentage value.
-19027	MESSAGE	Invalid Maximum CPU Percentage value.
	CAUSE	The value is less than minimum CPU Percentage.
	ACTION	Pass an appropriate value within the correct range for Maximum CPU Percentage value.
-19028	MESSAGE	Invalid Maximum CPU Percentage value.
	CAUSE	The value causes total percentage to be more than 100.
	ACTION	Pass an appropriate value for Maximum CPU Percentage value.
-19029	MESSAGE	Invalid pin number.
	CAUSE	A non-existent pin's workgroup cannot be changed.
	ACTION	Pass a valid pin number.
-19030	MESSAGE	Invalid workgroup name.
	CAUSE	A pin cannot be moved from a non-existent workgroup.
	ACTION	Pass a valid workgroup name.
-19031	MESSAGE	Invalid workgroup name.
	CAUSE	AS_DEFAULT or BS_DEFAULT workgroups cannot be modified.
	ACTION	Pass a valid workgroup name.
-19032	MESSAGE	Invalid workgroup file.
	CAUSE	The workgroup configuration file is corrupted.
	ACTION	The file will be regenerated at the next reboot.

-19033	MESSAGE	Invalid workgroup name.
	CAUSE	A pin cannot be pegged to a non-existent workgroup.
	ACTION	Pass a valid workgroup name.
-19034	MESSAGE	Base or limit priority not passed to AIFWGADD.
	CAUSE	Base and limit priority are required for addition of a new workgroup.
	ACTION	Pass appropriate values for base and limit.
-19035	MESSAGE	An internal error.
	CAUSE	An error occured while parsing of the file.
	ACTION	Check the file syntax and semantics before passing it to AIFWGREPLACE.
-19036	MESSAGE	Invalid workgroup name.
	CAUSE	Duplicate workgroup name.
	ACTION	Pass a valid workgroup name.
-19037	MESSAGE	Invalid workgroup name.
	CAUSE	The name of the workgroup passed is not valid.
	ACTION	Check the workgroup name and pass a valid workgroup name.
-19038	MESSAGE	Pathname not absolute.
	CAUSE	Pathname should be absolute.
	ACTION	Check the pathname and pass a valid pathname.
-19039	MESSAGE	Invalid indirect file passed to AIFWGREPLACE.
	CAUSE	Indirect file should be an ASCII file.
	ACTION	Check the file and pass an ASCII file's number.
-19040	MESSAGE	Workgroup name is reserved.
	CAUSE	A reserved workgroup name was passed.
	ACTION	Check the name and pass a valid workgroup name.

-19041	MESSAGE	Indirect file passed to AIFWGREPLACE cannot be processed.
	CAUSE	There is an unexpected system problem.
	ACTION	Contact Hewlett-Packard for support.
-19042	MESSAGE	Workload Manager not purchased.
	CAUSE	Workload Manager product should be purchased.
	ACTION	Contact Hewlett-Packard.
-19043	MESSAGE	Internal error.
	CAUSE	There is an unexpected system problem.
	ACTION	Contact Hewlett-Packard for support.
-19044	MESSAGE	Invalid item number passed in itemnum_array to AIFWGGET.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with zero.
-19045	MESSAGE	Invalid item number passed in ver_item_array to AIFWGPUT.
	CAUSE	A non-zero invalid item number was passed in ver_item_array.
	ACTION	Pass an appropriate value and end with zero.
-19046	MESSAGE	Invalid item number passed in itemnum_array to AIFWGPUT.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with zero.
-19047	MESSAGE	Invalid item number passed in itemnum_array to AIFWGADD.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with zero.
-19048	MESSAGE	Invalid item number passed in itemnum_array to AIFWGREPLACE.
	CAUSE	A non-zero invalid item number was passed in itemnum_array.
	ACTION	Pass an appropriate value and end with zero.
-19049	MESSAGE	Invalid workgroup name passed in to AIFWGGET or AIFWGPUT.
	CAUSE	A non-existent workgroup name was passed.
	ACTION	Check the workgroup name.

-19050	MESSAGE	Invalid MPE/iX priority value.
	CAUSE	MPE/iX priority value should be in the range of 127 to 13567.
	ACTION	Pass a priority value within the range.
-19051	MESSAGE	Invalid Quantum value.
	CAUSE	Quantum should be in the range of 1..32767.
	ACTION	Pass a quantum value within the range.
-19052	MESSAGE	Invalid Timeslice value.
	CAUSE	Timeslice is a multiple of 100 milliseconds and has a minimum value of 100 milliseconds.
	ACTION	Pass an appropriate value within the correct range for Timeslice value.
-19053	MESSAGE	Invalid Boost Property value.
	CAUSE	Boost property should be 0 for DECAY or 1 for OSCILLATE.
	ACTION	Pass an appropriate value within the correct range for Boost Property value.
-19054	MESSAGE	Invalid CPU percentage value.
	CAUSE	The value can range from 0% to 100%.
	ACTION	Pass an appropriate value within the correct range for CPU Percentage value.
-19055	MESSAGE	Minimum characteristics not passed.
	CAUSE	One of the membership criteria and the required scheduling characteristics of base and limit must be passed.
	ACTION	Pass one of the membership criteria and the required scheduling characteristics of base and limit.
-19056	MESSAGE	Array could not be processed.
	CAUSE	Arrays should be null terminated.
	ACTION	Null terminate the array and restart the AIF call.
-19057	MESSAGE	An attempt was made to change percentages of default workgroups.
	CAUSE	Percentages of default workgroups cannot be changed.
	ACTION	Check the AIF call and restart it.

B

AIF Data Structures

```
bit1                    =     0 .. 1;

bit2                    =     0 .. 3;

bit8                    =     0 .. 255;

bit14                   =     0 .. 16383;

bit16                   =     0 .. 65535;

bit31                   =     0 .. 2147483647;
```

```
buffer_type =

          record
              case boolean of
                true:   (buff_str : string[n]);
                false:  (buff_len : integer;
                         buffer   : packed array [1..n]);
              end;

Array size: 4 + n + 1 bytes

n represents a user-defined size.
```

```
buffer_info_type =

         record
         buffer_offset  : integer; (0.0 @ 4.0)
         pathname_len   : integer; (4.0 @ 4.0)
         end;
Record size: 8 bytes
Alignment:   4 bytes
```

```
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  :  4 bytes

The values for the user, group, and account must be upper-case.
```

```
drives_type =

record
  cnt          : integer;
  drives_ldevs : array [1..n] of integer;
end;

n represents a user-defined size.
Record size : 4 + 4n bytes
Alignment   : 4 bytes
```

```
 dstsrec_type  =

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


Record size: 12n + 4 bytes
Alignment  : 4 bytes

n represents 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.
```

```
 fnamerec_type       = record
                       cnt       : integer
                       fnames    : array[1..n] of
                                     filename_type;
                       end;

Record size: 4 + 20n bytes
Alignment  : 4 bytes
```

```
 fnumpid_type  =

         record
           fnum    : integer;                              (0.0 @ 4.0)
           pid     : longint_type;                         (4.0 @ 8.0)
         end ;


Record size: 12 bytes
Alignment  : 4 bytes
```

```
 i32rec_type       = record
                     cnt      : integer;
                     ints     : array[1..n] of integer;
                     end;

Record size: 4 + 4n bytes
Alignment  : 4 bytes
```

```
 i64rec_type      = record
                     cnt      : integer;
                     lints    : array[1..n] of longint;
                     end;

Record size: 4 + 8n bytes
Alignment  : 4 bytes
```

```
 item_array_type  =

       array[1..n] of globalanypointer;

Array size: 8n bytes
Element size: 8 bytes
Alignment: 4 bytes

n represents a user defined size.
```

```
 itemstatus_array_type  =

       array[1..n] of status_type;

Array size: 4n bytes
Element size: 4 bytes
Alignment:  4 bytes

n represents a user defined size.
```

```
 itemnum_array_type  =

       array[1..n] of integer;

Array size: 4n bytes
Element size: 4 bytes
Alignment:  4 bytes

n represents a user defined size.
```

```
 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  : 4 bytes

The js_type field can have a value of 1 or 2; 1 indicates a session
and 2 indicates a job. The js_num field can have 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
```

```
 max_pathlen =  1024

This value can be retrieved from AIFSCGET, item 3062.
```

```
 max_pathname_type =

         packed array [ 1 .. max_pathlen ] of char;

 Array size: max_pathlen bytes   Element Size: 1 byte
 Alignment: 1 byte
```

```
  message_buffer_type   =

         packed array[1..n] of char;

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

 The type is user-defined. n represents any number <= 32767.
```

```
media_label_type =

record
  media_name : packed array [1..32] of char;      (0.0 @ 20.0)
  subname1   : packed array [1..16] of char;     (20.0 @ 10.0)
  subname2   : packed array [1..16] of char;     (20.0 @ 10.0)
end;

Record size : 64 bytes
Alignment   : 1 byte
```

```
mm_side_type =

record
  media_label  : media_label_type;               (0.0 @ 40.0)
  volume_label : packed array[1..8] of char;    (40.0 @ 8.0)
end;

Record size : 72 bytes
Alignment   : 1 byte
```

```
mm_slot_info_type =

record
  slot_number : integer;                 (0.0 @ 4.0)
  slot_state  : mm_slot_state_type;      (4.0 @ 1.0)
  side_a      : mm_side_type;            (5.0 @ 48.0)
  side_b      : mm_side_type;           (4D.0 @ 48.0)
end;

Record size : 152 bytes
Alignment   : 4 bytes
```

```
mm_slot_state_type =

  (empty_slot, full_slot, rsvd_slot);
```

```
  mpe_name_type      =

      packed array[1..16] of char;       (0.0 @ 16)

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

```
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
```

```
path_identifier   =  $alignment 4$

          record
               ufid        :  ufid_type;
               link_id     :  bit32;
               parent_ufid :  ufid_type;
          end;

 Record size: 44 bytes
 Alignment : 4 bytes
```

```
path_id_rec_type   =  $alignment 4$

          record
               cnt         :  integer;
               path_ids    :  rec [1..n] of path_identifier;
          end;

 n represents a user-defined size.
 Record size: 4 + 44n bytes
 Alignment : 4 bytes
```

```
pathname_type =

          record
              case boolean of
                 true:  (path_str :  string [n]);
                 false: (path_rec :  record
                           length      :  integer;
                           pathname    :  packed array [1..n] of char;
                           end);
                 end;


 Record size : 4 + n + 1 bytes
 Alignment   : 4 bytes

 n represents a user-defined size.    Currently the maximum pathname
 length is 1024 bytes.
```

```
 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..n] of fnumpid_type               (4.0 @ 12n.0)
          end ;

 Record size: 12n + 4 bytes
 Alignment  : 4 bytes

 n represents 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_ufid    : ufid_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)
             12: ( key_ldev    : integer );                  (0.0 @ 4.0)
             13: ( key_va      : bit64   );                  (0.0 @ 8.0)
             14: ( key_int     : integer );                  (0.0 @ 4.0)
             15: ( key_class   : pac16   );                  (0.0 @ 16.0)
             16: ( key_pathname: max_pathname_type );        (0.0 @ 1024.0)
         end ;

   Record size: 1024 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
```

```
storage_slot_type =

record
  lower_limit  : integer;
  upper_limit  : integer;
  slot_info    : array [1..n] of mm_slot_info_type;
end;

n represents a user-defined size.
Record size : 4 + 4 + 152n
Alignment   : 4 bytes
```

```
t_vol_class_name =

     packed array[1..32] of char;                          (0.0 @ 32.0)

Array size : 32 bytes
Alignment  : 1  byte
```

```
t_volume_name    =

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

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

```
t_vol_set_name   =

     packed array[1..32] of char;                         (0.0 @ 32.0)
```

```
ufidrec_type           = record
                           cnt          : integer;
                           ufids        : array[1..n] of
                                          ufid_type;
                           end;
```

```
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
```

```
key_workgroup_type =
record
 wgindex           : integer;            (0.0 @ 4.0)
 creation_count    : integer;            (4.0 @ 4.0)
end;

Record Size  : 8 bytes
Alignment    : 4 bytes
```

```
pac20 =

   packed array[1..20] of char;          (0.0 @ 14.0)

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

C

Programming Examples

This appendix contains programming examples to illustrate the use of operating system architected interfaces.

Examples one and two illustrate using AIF:OS system calls to send and receive data. These two examples are written in HP C/iX.

Examples three and four illustrate using AIF:OS system calls with asynchronous ports. These two examples are written in HP Pascal.

Example five illustrates using `AIFSYSWIDEGET` for retrieving HFS pathnames. This example is written in HP Pascal.

Example six illustrates using `AIFSYSWIDEGET` for traversing HFS directories.

Example seven illustrates using the Magneto-Optical AIF's. This example is written in HP Pascal.

Example 1 - SEND1S, send data

Examples one and two illustrate using the AIF:OS system calls to send and receive data. These two examples are written in HP C/iX.

To compile the source code for SEND1S:

```
ccxl send1s,,$null;info="-Aa +e"
link from=$oldpass,xdbend.lib.sys;to=send1p;&
             cap=ia,pm;rl=libcinit.lib.sys
```

This is the source code for program SEND1S:

```
#pragma list off
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <mpe.h>
#pragma list on

#pragma intrinsic_file "aifintr.pub.sys"
#pragma intrinsic      AIFPORTOPEN
#pragma intrinsic      AIFPORTCLOSE
#pragma intrinsic      AIFPORTSEND

#pragma intrinsic_file "sysintr.pub.sys"
#pragma intrinsic      GETPRIVMODE
#pragma intrinsic      GETUSERMODE

#define num_items      50
typedef int^ item_array_type[num_items];
typedef int  itemnum_array_type[num_items];
typedef t_mpe_status  item_status_array_type[num_items];

/*
This program SEND1S and its counterpart RECV1S will use AIF:OS intrinsics
to send and receive data.
*/
   static item_array_type          item_array;
   static itemnum_array_type       itemnum_array;
   static item_status_array_type   itemstatus_array;
   static t_mpe_status             overall_status;
   static char                     error_stng[80];
   static int                      user_id = 123456789; /* Your valid user_id */
   static int                      port_id = 0;
```

```
/*************************************************************************/
/*                          fatal_aif_error                              */
/*************************************************************************/
void fatal_aif_error (t_mpe_status             *overall status
                      char
                      item_status_array_type
                      item_status_array)
{
  int i;

  printf ("**** %s\n", error_stng);
  printf ("**** error: Info=%d Subsystem=%d\n",
               overall_status->decode.error_num
               overall_status->decode.subsys_num
  for (i=0; i, num_items; i++|) {
     if (item_status_array[i].decode.error.num) {
       printf("\n Loop: %d error: Info=%d Subsystem=%d\n",
              i, item_status_array[i].decode.error_num,
              item_status_array[i].decode.subsys_num);
        } /* if */
     }     /* for loop */
  QUIT (1);
} /* fatal_aif_error */
```

```
/*************************************************************************/
/*                          open_port                                    */
/*************************************************************************/
void open_port (int    *port_id)
{
   char port_name[16]     = "PORT2          ";
   char port_password[16] = "PORT2_PASS     ";
   int  access_mode       = 2;
   int  create_option     = 1; /* default, but we're showing an example */

   itemnum_array[0] = 11201;
   item_array[0]    = &create_option;
   itemnum_array[1] = 0;
   GETPRIVMODE ();
   *port_id = AIFPORTOPEN (&overall_status, port_name, port_password,
                           access_mode, user_id,
                           itemnum_array, item_array, itemstatus_array);
   GETUSERMODE ();
   if (overall_status.word) {
      strcpy (error_stng, "AIFPORTOPEN intrinsic error");
      fatal_aif_error (&overall_status, error_stng, itemstatus_array);
      }
} /* open_port */

/*************************************************************************/
/*                          close_port                                   */
/*************************************************************************/
void close_port (int    *port_id)
{
   int  access_mode       = 2;

   GETPRIVMODE ();
   AIFPORTCLOSE (&overall_status, port_id, access_mode);
   GETUSERMODE ();
   if (overall_status.word) {
      strcpy (error_stng, "AIFPORTCLOSE intrinsic error");
      fatal_aif_error (&overall_status, error_stng, itemstatus_array);
      }
} /* close_port */
```

```
/*************************************************************************/
/*                          send_stng                                    */
/*************************************************************************/
void send_stng (int    *port_id,
                char   *stng,
                int    msg_pri)
{
   int      stng_len;
   stng_len = strlen (stng);
   itemnum_array[0] = 11102;
   item_array[0]    = &msg_pri;  /* set the pri*/
   itemnum_array[1] = 0;
   printf ("\nSending msg: %s: to port: %d  at priority %d\n",
           stng, *port_id, msg_pri);
   GETPRIVMODE ();
   AIFPORTSEND (&overall_status, port_id, stng, stng_len
                , ,, itemnum_array, item_array, itemstatus_array);
   GETUSERMODE ();
   if (overall_status.word) {
      strcpy (error_stng, "AIFPORTSEND  intrinsic error");
      fatal_aif_error (&overall_status, error_stng, itemstatus_array);
      }
} /* send_stng */


/*************************************************************************/
/*                                    MAIN                               */
/*************************************************************************/
int main(int argc, char *argv[], char *envp[], int parm)
{
   char                          stng[80];
   char                          pri_stng[20];
   int                           msg_pri;

   open_port (&port_id);

   printf ("\n\nEnter message to be sent (Q to quit) : ");
   gets (stng);

   while (strncmp (stng, "Q", 1) != 0) {
      printf ("\n\nEnter message priority : ");
      gets (pri_stng);
      msg_pri = atoi (pri_stng);
      send_stng (&port_id, stng, msg_pri);
      printf ("\n\nEnter message to be sent (Q to quit) : ");
      gets (stng);
      }

   close_port (&port_id);
}
```

Example 2 - RECV1S, receive data

The program RECV1S and its counterpart SEND1S use the AIF:OS system calls to send and receive the data.

This is the source code for program RECV1S:

```
#pragma list off
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <mpe.h>
#pragma list on

#pragma intrinsic_file "aifintr.pub.sys"
#pragma intrinsic      AIFPORTOPEN
#pragma intrinsic      AIFPORTCLOSE
#pragma intrinsic      AIFPORTSEND

#pragma intrinsic_file "sysintr.pub.sys"
#pragma intrinsic      GETPRIVMODE
#pragma intrinsic      GETUSERMODE

#define num_items      50
typedef int^ item_array_type[num_items];
typedef int  itemnum_array_type[num_items];
typedef t_mpe_status  item_status_array_type[num_items];

/*
This program RECV1s and its counterpart SEND1S will use AIF:OS intrinsics
to send and receive data.
*/

   static item_array_type          item_array;
   static itemnum_array_type       itemnum_array;
   static item_status_array_type   itemstatus_array;
   static t_mpe_status             overall_status;
   static char                     error_stng[80];
   static int                      user_id = 123456789; /* your valid user_id */
   static int                      port_id = 0;
```

```
/*************************************************************************/
/*                          fatal_aif_error                              */
/*************************************************************************/
void fatal_aif_error (t_mpe_status             *overall status
                      char
                      item_status_array_type
                      item_status_array)
{
  int i;

  printf ("**** %s\n", error_stng);
  printf ("**** error: Info=%d Subsystem=%d\n",
               overall_status->decode.error_num
               overall_status->decode.subsys_num
  for (i=0; i, num_items; i++|) {
     if (item_status_array[i].decode.error.num) {
       printf("\n Loop: %d error: Info=%d Subsystem=%d\n",
              i, item_status_array[i].decode.error_num,
              item_status_array[i].decode.subsys_num);
         } /* if */
     }     /* for loop */
  QUIT (1);
} /* fatal_aif_error */
```

```
/*************************************************************************/
/*                          open_port                                    */
/*************************************************************************/
void open_port (int    *port_id)
{
   char port_name[16]     = "PORT1           ";
   char port_password[16] = "PORT1_PASS      ";
   int  access_mode       = 1;
   int  create_option     = 1; /* default, but we're showing an example */

   itemnum_array[0] = 11201;
   item_array[0]    = &create_option;
   itemnum_array[1] = 0;
   GETPRIVMODE ();
   *port_id = AIFPORTOPEN (&overall_status, port_name, port_password,
                           access_mode, user_id,
                           itemnum_array, item_array, itemstatus_array);
   GETUSERMODE ();
   if (overall_status.word) {
      strcpy (error_stng, "AIFPORTOPEN intrinsic error");
      fatal_aif_error (&overall_status, error_stng, itemstatus_array);
      }
} /* open_port */



/*************************************************************************/
/*                          close_port                                   */
/*************************************************************************/
void close_port (int    *port_id)
{
   int  access_mode       = 1;

   GETPRIVMODE ();
   AIFPORTCLOSE (&overall_status, port_id, access_mode);
   GETUSERMODE ();
   if (overall_status.word) {
      strcpy (error_stng, "AIFPORTCLOSE intrinsic error");
      fatal_aif_error (&overall_status, error_stng, itemstatus_array);
      }
} /* close_port */
```

```
/*************************************************************************/
/*                          receive_stng                                 */
/*************************************************************************/
int  receive_stng (int    *port_id,
                   char   *stng,
                   int    stng_len,
                   int    msg_pri)
{
   int      time_out = -1;

   itemnum_array[0] = 11002;
   item_array[0]    = &time_out;
   itemnum_array[1] = 11001;
   item_array[1]    = &msg_pri;
   itemnum_array[2] = 0;
   GETPRIVMODE ();
   printf ("\nChecking for msg at pri: %d\n", msg_pri);
   AIFPORTRECEIVE (&overall_status, port_id, stng, &stng_len
                   , ,, itemnum_array, item_array, itemstatus_array);
   GETUSERMODE ();
   if (( (overall_status.decode.error_num == -111) ||
         (overall_status.decode.error_num == -129)) &&&
       (overall_status.decode.subsys_num == 516)) {
       printf ("No message found at that priority\n");
       return FALSE;
       }
   else if (overall_status.word) {
      strcpy (error_stng, "AIFPORTRECEIVE intrinsic error");
      fatal_aif_error (&overall_status, error_stng, itemstatus_array);
      }
   else
      stng[stng_len] = '\0';
      return TRUE;
} /* receive_stng */
```

```
/*************************************************************************/
/*                                  MAIN                                 */
/*************************************************************************/
int main(int argc, char *argv[], char *envp[], int parm)
{
   char                          stng[80];
   int                           msg_pri;

   open_port (&port_id);

   printf ("\n\nEnter message priority (0 to quit) : ");
   scanf ("%d", &msg_pri);

   printf ("portid is %d\n", port_id);
   while (msg_pri) {
      if (receive_stng (&port_id, stng, sizeof (stng), msg_pri)) {
         printf ("\nReceived msg: %s  at pri: %d\n", stng, msg_pri);
         }
      printf ("\n\nEnter message priority : ");
      scanf ("%d", &msg_pri);
      }

   close_port (&port_id);
}
```

Example 3 - ASYNC1, asynchronous ports

```
$standard_level 'ext_modcal'$
{-------------------------------------------------------------------------
   program ASYNC1
-------------------------------------------------------------------------

PURPOSE:

This is a simple program to illustrate the use of asynchronous ports.

1.  Call GETPRIVMODE to gain user privilege level 2.
2.  Create and open an asynchronous port using AIFPORTOPEN.
3.  Create and open a synchronous port using AIFPORTOPEN.
4.  Send message with AIFPORTSEND to notify the other process the
    asynchronous port exists.
5.  Pause to wait for a message.
6.  The other program will send multiple messages to the asynchronous
    port which will cause the interrupt handler to run.
7.  Use AIFPORTINT to disable/enable interrupts in the handler.
8.  Call AIFPORTRECEIVE to receive the message in the handler.
9.  Use AIFPORTCLOSE and close each port.
10. Call GETUSERMODE.

PARAMETERS: None.

-------------------------------------------------------------------------}

program ASYNC1 (input,output);


type

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

   bit32                  = packed array [1..32] of boolean;
   item_array_type        = array [1..5] of globalanyptr;
   itemnum_array_type     = array [1..5] of integer;
   item_status_array_type = array [1..5] of status_type;
   message_buffer_type    = packed array [1..80] of char;
   name_type              = packed array [1..16] of char;
```

```
{-------------------------------------------------------------------------
  declare structured constants to initialize arrays used in various
  AIF procedure calls
-------------------------------------------------------------------------}

const
   Init_Item_Array        = item_array_type
                               [5 of nil];

   Init_Itemnum_Array     = itemnum_array_type
                               [5 of 0];

   Init_Item_Status_Array = item_status_array_type
                               [5 of status_type
                                  [info   : 0,
                                   subsys : 0]];

var
   accessmode             : integer;
   createoptions          : integer;
   createhandler          : bit32;
   createstate            : boolean;
   envelope_code          : integer;
   interval               : real;
   itemnum_ports          : itemnum_array_type;
   item_ports             : item_array_type;
   item_status_ports      : item_status_array_type;
   maxmsgsize             : integer;
   message_buffer         : message_buffer_type;
   message_id             : integer;
   message_length         : integer;
   newstates              : array [1..2] of boolean;
   overall_status         : status_type;
   portid1                : integer;
   portid2                : integer;
   portlist               : array [1..2] of integer;
   oldstates              : array [1..2] of integer;
   portname               : name_type;
   portpass               : name_type;
   proc_name              : packed array [1..13] of char;
   proc_file              : packed array [1..8] of char;
   user_id                : integer;
   timeoutseconds         : integer;
```

```
procedure GETPRIVMODE;    intrinsic;
procedure GETUSERMODE;    intrinsic;
procedure HPGETPROCPLABEL; intrinsic;
procedure PAUSE;          intrinsic;
procedure QUIT;           intrinsic;

$sysintr 'aifintr'$
procedure AIFPORTCLOSE;   intrinsic;
procedure AIFPORTINT;     intrinsic;
function  AIFPORTOPEN   : integer; intrinsic;
procedure AIFPORTRECEIVE; intrinsic;
procedure AIFPORTSEND;    intrinsic;

{------------------------------------------------------------------------
   procedure ERROR_IN_CALL
------------------------------------------------------------------------

   PURPOSE:
     This procedure will accept an intrinsic call name and status
     variable. It will output a message naming the offending call
     and status information and subsystem parameters. It will call
     QUIT to terminate the calling program and child processes.

   PARAMETERS:
    name   - name of erroring AIF or intrinsic call
    status - status of call
    item_status_array - array of status values for item list

------------------------------------------------------------------------}
procedure ERROR_IN_CALL (     name              : name_type;
                              status            : status_type;
                         var item_status_array : item_status_array_type)
option extensible 2;

var
   index   : shortint;
   quitnum : integer;

begin { ERROR_IN_CALL }

   writeln ('Error in ',name);
   writeln ('Overall status info =',status.info, ' subsys =',
             status.subsys);
   if status.info > 0 then
      for index := 1 to status.info do
         writeln ('Index: ',index,' info =',item_status_array[index].info,
                  ' subsys =',item_status_array[index].subsys);
   quitnum := 516;
   QUIT (quitnum);

end; { ERROR_IN_CALL }
```

```
{-------------------------------------------------------------------------
   procedure PORTHANDLER
-------------------------------------------------------------------------

   PURPOSE:
   This procedure is the handler for the asynchronous port. It
   illustrates the use of AIFPORTINT to disable port interrupts.
   It will use AIFPORTRECEIVE to get the message. It will reissue
   the AIFPORTRECEIVE to check if messages with pending interrupts
   have arrived. AIFPORTINT will enable port interrupts before
   exiting the handler.

   PARAMETERS:
   portid - contains the portid of the port with the incoming message
            this is a required parameter
-------------------------------------------------------------------------}
procedure PORTHANDLER ( portid : integer);

var
  env_code          : integer;
  hand_status       : status_type;
  index             : integer;
  item_porth        : item_array_type;
  item_status_porth : item_status_array_type;
  itemnum_porth     : itemnum_array_type;
  msg_buf           : message_buffer_type;
  msg_id            : integer;
  msg_len           : integer;
  newstates         : array[1..2] of boolean;
  oldstates         : array[1..2] of boolean;
  pending           : boolean;
  portlist          : array[1..2] of integer;
  timeout           : integer;

begin { PORTHANDLER }
```

```
{-------------------------------------------------------------------------
 7. AIFPORTINT is setup to disable interrupt handling on the current port.
    This is done when an application is accessing global data or has a
    critical section of code to protect.

    It is not necessary to call AIFPORTINT inside a handler. If new messages
    arrive, the interrupt handlers will nest. It is recommended the
    processing done in the handler be kept at a minimum.

-------------------------------------------------------------------------}
hand_status.all := 0;
newstates[1]    := False;
portlist[1]     := portid;
portlist[2]     := 0;
AIFPORTINT ( hand_status,
             portlist,
             newstates,
             oldstates);
if hand_status.all  0 then
   writeln ('AIFPORTINT 1 - Bad Status: info =',hand_status.info,
            ' subsys =', hand_status.subsys)
else
  if oldstates[1] then
    writeln('AIFPORTINT 1 (handler) - Previous state of port is ENABLED.')
  else
    writeln('AIFPORTINT 1 (handler) -  Previous state of port is DISABLED.');
```

```
{-------------------------------------------------------------------------
8.  AIFPORTRECEIVE is called to get the original message which caused
    the handler to be invoked. Variables should be initialized before
    the AIFPORTRECEIVE call.
-------------------------------------------------------------------------}
  msg_len           := 80;
  env_code          := 0;
  msg_id            := 0;
  msg_buf           := ' ';
  itemnum_porth     := Init_Itemnum_Array;        { zero array }
  item_porth        := Init_Item_Array;
  item_status_porth := Init_Item_Status_Array;
  timeout           := -1;                        { nowait receive }
  itemnum_porth[1]  := 11002;              { next element initialized to 0 }
  item_porth[1]     := addr(timeout);

  AIFPORTRECEIVE ( hand_status,
                   portid,
                   msg_buf,
                   msg_len,
                   env_code,
                   msg_id,
                   itemnum_porth,
                   item_porth,
                   item_status_porth);

  if hand_status.all  0 then
       begin
       writeln ('AIFPORTRECEIVE - Bad Status: info=' ,
                hand_status.info, ' subsys =', hand_status.subsys);
       if hand_status.info > 0 then
         for index := 1 to hand_status.info do
         writeln ('Index: ',index,' info =',item_status_porth[index].info,
                  ' subsys =',item_status_porth[index].subsys);
       end
  else
    writeln('AIFPORTRECEIVE (complete) - ' , msg_buf );
```

```
{-------------------------------------------------------------------------
 The AIFPORTRECEIVE can be used to check if there are messages with
 pending interrupts. If the message is successfully received the
 pending interrupt will not occur when item 11007 is used.
-------------------------------------------------------------------------}

  msg_len           := 80;
  env_code          := 0;
  msg_id            := 0;
  msg_buf           := ' ';
  itemnum_porth     := Init_Itemnum_Array;        { zero array }
  item_porth        := Init_Item_Array;
  item_status_porth := Init_Item_Status_Array;
  timeout           := -1;                        { nowait receive }
  itemnum_porth[1]  := 11002;
  item_porth[1]     := addr(timeout);
  pending           := True;
  itemnum_porth[2]  := 11007;              { next element initialized to 0 }
  item_porth[2]     := addr(pending);

  while pending do
  begin  { while pending }
    AIFPORTRECEIVE ( hand_status,
                     portid,
                     msg_buf,
                     msg_len,
                     env_code,
                     msg_id,
                     itemnum_porth,
                     item_porth,
                     item_status_porth);

    if hand_status.all  0 then
         begin
         writeln ('AIFPORTRECEIVE - Bad Status: info=' ,
                  hand_status.info, ' subsys =', hand_status.subsys);
         if hand_status.info > 0 then
           for index := 1 to hand_status.info do
             writeln ('Index: ',index,' info =',item_status_porth[index].info,
                      ' subsys =',item_status_porth[index].subsys);
         pending := False;
         end
    else
      writeln('AIFPORTRECEIVE (pending) - ' , msg_buf );
  end;   { while pending }
```

```
{-------------------------------------------------------------------------
 Call AIFPORTINT to enable/arm interrupt handling
-------------------------------------------------------------------------}

newstates[1] := oldstates[1];
AIFPORTINT ( hand_status,
             portlist,
             newstates,
             oldstates);
if hand_status.all  0 then
   writeln ('AIFPORTINT 2 - Bad Status: info =',hand_status.info,
            ' subsys =', hand_status.subsys)
else
  if oldstates[1] then
    writeln('AIFPORTINT 2 (handler) - Previous state of port is ENABLED.')
  else
    writeln('AIFPORTINT 2 (handler) -  Previous state of port is DISABLED.');

end; { PORTHANDLER }

begin {ASYNC1}

{-------------------------------------------------------------------------
1.  Call GETPRIVMODE to gain user privilege level 2
-------------------------------------------------------------------------}
GETPRIVMODE;
```

```
{-------------------------------------------------------------------------
2.  Create and open an asynchronous port using AIFPORTOPEN.
-------------------------------------------------------------------------}
writeln('Enter a valid user id:');
readln(user_id);

portname          := 'aifport1        ';
portpass          := 'aifpass1        ';
accessmode        := 1;                    { receive access }
itemnum_ports     := Init_Itemnum_Array;   { zero array }
item_ports        := Init_Item_Array;
item_status_ports := Init_Item_Status_Array;
createoptions     := 2;                    { create new     }
itemnum_ports [1] := 11201;
item_ports [1]    := addr(createoptions);
maxmsgsize        := 80;                   { message size   }
itemnum_ports [2] := 11202;
item_ports [2]    := addr(maxmsgsize);
proc_name         := '#PORTHANDLER#';
proc_file         := '#ASYNC1#';
HPGETPROCPLABEL   (proc_name,
                   createhandler,
                   overall_status,
                   proc_file,
                   False);
if overall_status.all  0 then
   ERROR_IN_CALL('HPGETPROCPLABEL',overall_status);
itemnum_ports [3] := 11206;
item_ports [3]    := addr(createhandler);        { handler address }
createstate       := True;
itemnum_ports [4] := 11207;               { next element initialized to 0 }
item_ports [4]    := addr(createstate);          { handler enabled   }

portid1 := AIFPORTOPEN(overall_status,
                       portname,
                       portpass,
                       accessmode,
                       user_id,
                       itemnum_ports,
                       item_ports,
                       item_status_ports);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTOPEN',overall_status,item_status_ports);
```

```
{-------------------------------------------------------------------------
3.  Create and open a synchronous port using AIFPORTOPEN.
-------------------------------------------------------------------------}
portname          := 'aifport2       ';
portpass          := 'aifpass2       ';
accessmode        := 2;                    { send access    }
itemnum_ports [3] := 0;                    { terminate item list }
portid2 := AIFPORTOPEN(overall_status,
                       portname,
                       portpass,
                       accessmode,
                       user_id,
                       itemnum_ports,
                       item_ports,
                       item_status_ports);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTOPEN #2',overall_status,item_status_ports);
```

```
{-------------------------------------------------------------------------
4.  Send message with AIFPORTSEND to notify the other process the
    asynchronous port exists. Once the asynchronous port exists the
    other process can open the port. The asynchronous port creator
    is the only receiver and must be the first opener.
-------------------------------------------------------------------------}

message_buffer    := 'Asynchronous port open OK to send messages ';
message_length    := 80;
itemnum_ports     := Init_Itemnum_Array;         { zero array }
item_ports        := Init_Item_Array;
item_status_ports := Init_Item_Status_Array;
timeoutseconds    := 0; { wait for other process to receive message }
item_ports[1]     := ADDR(timeoutseconds);
itemnum_ports[1]  := 11101;         { next element initialized to 0 }
AIFPORTSEND (overall_status,
             portid2,
             message_buffer,
             message_length,
             ,
             ,
             itemnum_ports,
             item_ports,
             item_status_ports);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTSEND',overall_status,item_status_ports);

{-------------------------------------------------------------------------
5.  Pause to wait for a message. In a true application there would be
    real processing taking place. Otherwise, a synchronous port should
    be used if there is no work for the application to do.
6.  The other program will send a message which will cause the interrupt
    handler to run.
-------------------------------------------------------------------------}
interval := 60.0;
PAUSE (interval);
```

```
{------------------------------------------------------------------------
9.  Close the ports.
------------------------------------------------------------------------}
AIFPORTCLOSE ( portid2,
               accessmode,
               overall_status);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTCLOSE',overall_status);
AIFPORTCLOSE ( portid1,
               accessmode,
               overall_status);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTCLOSE #2',overall_status);

{------------------------------------------------------------------------
10. Call GETUSERMODE
------------------------------------------------------------------------}
GETUSERMODE;
writeln ('ASYNC1 terminated')
end.
```

Example 4 - ASYNC2, asynchronous ports

```
$standard_level 'ext_modcal'$
{-------------------------------------------------------------------------
   program ASYNC2
-------------------------------------------------------------------------

PURPOSE:

This is a simple program to illustrate the use of asynchronous ports.

1.  Call GETPRIVMODE to gain user privilege level 2
2.  Open a synchronous port named 'aifport2' for receive access.
3.  Call AIFPORTRECEIVE on 'aifport2' to wait for message to open
    asynchronous port 'aifport1'.
4.  After message arrives, AIFPORTOPEN the asynchronous port named
    'aifport1' for send access.
5.  Use AIFPORTSEND to send multiple messages to 'aifport1'.
6.  Close the ports.
7.  Call GETUSERMODE.

PARAMETERS: None.

-------------------------------------------------------------------------}

program ASYNC2 (input,output);

type

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

   item_array_type        = array [1..3] of globalanyptr;
   itemnum_array_type     = array [1..3] of integer;
   item_status_array_type = array [1..3] of status_type;
   name_type              = packed array [1..16] of char;
   message_buffer_type    = packed array [1..80] of char;
```

```
{-------------------------------------------------------------------------
  declare structured constants to initialize arrays used in various
  AIF procedure calls
-------------------------------------------------------------------------}

const
   Init_Item_Array        = item_array_type
                               [3 of nil];

   Init_Itemnum_Array     = itemnum_array_type
                               [3 of 0];

   Init_Item_Status_Array = item_status_array_type
                               [3 of status_type
                                  [info   : 0,
                                   subsys : 0]];

var
   accessmode             : integer;
   base                   : integer;
   count_string           : string[6];
   envelope_code          : integer;
   index                  : integer;
   interval               : real;
   itemnum_ports          : itemnum_array_type;
   item_ports             : item_array_type;
   item_status_ports      : item_status_array_type;
   maxmsgsize             : integer;
   message_buffer         : message_buffer_type;
   message_id             : integer;
   message_length         : integer;
   message_string         : string[40];
   numchar                : shortint;
   overall_status         : status_type;
   portid1                : integer;
   portid2                : integer;
   portname               : name_type;
   portpass               : name_type;
   timeoutseconds         : integer;
   user_id                : integer;
```

```
function  ASCII         : shortint; intrinsic;
procedure GETPRIVMODE;    intrinsic;
procedure GETUSERMODE;    intrinsic;
procedure PAUSE;          intrinsic;
procedure QUIT;           intrinsic;
$sysintr 'aifintr'$
procedure AIFPORTCLOSE;   intrinsic;
function  AIFPORTOPEN   : integer; intrinsic;
procedure AIFPORTRECEIVE; intrinsic;
procedure AIFPORTSEND;    intrinsic;

{------------------------------------------------------------------------
   procedure ERROR_IN_CALL
------------------------------------------------------------------------

   PURPOSE:
     This procedure will accept an intrinsic call name and status
     variable. It will output a message naming the offending call
     and status information and subsystem parameters. It will call
     QUIT to terminate the calling program and child processes.

   PARAMETERS:
    name   - name of erroring AIF or intrinsic call
    status - status of call
    item_status_array - array of status values for item list

------------------------------------------------------------------------}
procedure ERROR_IN_CALL (     name              : name_type;
                              status            : status_type;
                         var item_status_array : item_status_array_type)
option extensible 2;

var
   index   : shortint;
   quitnum : integer;

begin { ERROR_IN_CALL }
   writeln ('Error in ',name);
   writeln ('Overall status info =',status.info, ' subsys =',
             status.subsys);
   if status.info > 0 then
      for index := 1 to status.info do
         writeln ('Index: ',index,' info =',item_status_array[index].info,
                  ' subsys =',item_status_array[index].subsys);
   quitnum := 516;
   QUIT (quitnum);
end; { ERROR_IN_CALL }
```

```
begin    {ASYNC2 body}
{-------------------------------------------------------------------------
1.  Call GETPRIVMODE to gain user privilege level 2
-------------------------------------------------------------------------}
GETPRIVMODE;

{-------------------------------------------------------------------------
2.  Open a synchronous port named 'aifport2' for receive access.
-------------------------------------------------------------------------}
writeln ('Enter a valid user ID:');
readln(user_id);

portname          := 'aifport2        ';
portpass          := 'aifpass2        ';
accessmode        := 1;                    { receive access }
itemnum_ports     := Init_Itemnum_Array;   { zero array }
item_ports        := Init_Item_Array;
item_status_ports := Init_Item_Status_Array;
maxmsgsize        := 80;                   { message size   }
itemnum_ports [1] := 11202;                { next element initialized to 0 }
item_ports [1]    := addr(maxmsgsize);

portid2 := AIFPORTOPEN(overall_status,
                       portname,
                       portpass,
                       accessmode,
                       user_id,
                       itemnum_ports,
                       item_ports,
                       item_status_ports);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTOPEN #2',overall_status,item_status_ports);

{-------------------------------------------------------------------------
3.  Call AIFPORTRECEIVE on 'aifport2' to wait for message to open
    asynchronous port 'aifport1'. Defaults to waited receive.
-------------------------------------------------------------------------}
message_buffer    := ' ';
message_length    := 80;
itemnum_ports     := Init_Itemnum_Array;   { zero array }
item_ports        := Init_Item_Array;
item_status_ports := Init_Item_Status_Array;
AIFPORTRECEIVE ( overall_status,
                 portid2,
                 message_buffer,
                 message_length);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTRECEIVE',overall_status,item_status_ports);

writeln ('AIFPORTRECEIVE (complete) - ', message_buffer );
```

```
{-------------------------------------------------------------------------
4.  After message arrives, AIFPORTOPEN the asynchronous port named
    'aifport1' for send access.
-------------------------------------------------------------------------}
portname          := 'aifport1        ';
portpass          := 'aifpass1        ';
accessmode        := 2;                    { send access    }
user_id           := 555;
itemnum_ports     := Init_Itemnum_Array;   { zero array }
item_ports        := Init_Item_Array;
item_status_ports := Init_Item_Status_Array;
maxmsgsize        := 80;                   { message size   }
itemnum_ports [1] := 11202;                { next element initialized to 0 }
item_ports [1]    := addr(maxmsgsize);

portid1 := AIFPORTOPEN(overall_status,
                       portname,
                       portpass,
                       accessmode,
                       user_id,
                       itemnum_ports,
                       item_ports,
                       item_status_ports);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTOPEN #1',overall_status,item_status_ports);

for index := 1 to 32 do
  begin { for repeat send messages }
    message_buffer    := '';
    message_string    := '';
    base              := 10;
    numchar           := ASCII(index,base,count_string);
    SETSTRLEN(count_string,numchar);
    message_string    := 'AIFPORT1 send message # ' + count_string;
    STRMOVE (STRLEN(message_string),message_string,1,message_buffer,1);
    message_length    := 40;
    itemnum_ports     := Init_Itemnum_Array;   { zero array }
    item_ports        := Init_Item_Array;
    item_status_ports := Init_Item_Status_Array;
    timeoutseconds    := -1;            { nowait send }
    itemnum_ports[1]  := 11101;         { next element initialized to 0 }
    item_ports[1]     := ADDR(timeoutseconds);
    AIFPORTSEND       ( overall_status,
                        portid1,
                        message_buffer,
                        message_length,
                        ,
                        ,
                        itemnum_ports,
                        item_ports,
                        item_status_ports);
```

```
    if overall_status.all  0 then
      ERROR_IN_CALL('AIFPORTSEND',overall_status);
  end;  { for repeat send messages }

{-------------------------------------------------------------------------
6.  Close the ports.
-------------------------------------------------------------------------}
AIFPORTCLOSE ( portid2,
               accessmode,
               overall_status);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTCLOSE',overall_status);
AIFPORTCLOSE ( portid1,
               accessmode,
               overall_status);
if overall_status.all  0 then
   ERROR_IN_CALL('AIFPORTCLOSE #2',overall_status);

{-------------------------------------------------------------------------
7.  Call GETUSERMODE
-------------------------------------------------------------------------}
GETUSERMODE;
writeln ('ASYNC2 terminated')
end.
```

Example 5 - Retrieving HFS pathnames

Below is a sample program to illustrate the usage of AIFSYSWIDEGET for retrieving HFS pathnames. It will retrieve a list of DIRECTORY files while only traversing the first level of the directory. This will result in retrieving account names and HFS directories created at the root level. Specifying a traversal value of 0 is a much faster way of retrieving a list of accounts than searching the entire directory. This sample program will also call AIFFILEGGET with each pathname and retrieve the file owner.

Note that the program will test for specific errors from AIFSYSWIDEGET and only continue directory traversal for selected errors.

```
$standard_level 'ext_modcal'$
Program Sample( input, output);

Type

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

      buffer_info_type = record
                         buff_offset  : integer;
                         name_len     : integer;
                         end;

      buffer_type = record
                    case boolean of
                      true : (buff_str  : string[200000]);
                      false: (buff_len  : integer;
                              buffer    : packed array[1..200000] of char);
                    end;

      buffer_rec_ptr_type    = ^ $extnaddr$ buffer_type;

      item_status_array_type = array [ 1..5 ] of status_type;

      max_pathname_type      = packed array [ 1..1024 ] of char;

      name_type              = packed array [ 1..16 ] of char;

      ufid_type              = packed array [ 1..20 ] of char;
```

```
      pathname_type= record
                     case boolean of
                        true : (path_str :  string [1024]);
                        false: (length   :  integer;
                                pathname :  packed array [1..1024] of char);
                     end;

      path_identifier_type = $alignment 4$
                             record
                             ufid       : ufid_type;
                             link_id:      bit32;
                             parent_ufid:  ufid_type;
                             end;


      return_array1_type = array [1..1000] of path_identifier_type;

      return_array2_type = array [1..1000] of buffer_info_type;




Const
      blanks                 = max_pathname_type [1024 of ' '];

      init_item_status_array = item_status_array_type
                               [ 5 of status_type [all :0]];

      init_return_array1     = return_array1_type
                               [ 1000 of path_identifier_type
                                  [ufid       : '                    ',
                                   link_id    : 0,
                                   parent_ufid: '                    ']];

      null_chr               = chr(0);
```

```
Var
    access           : integer;
    answer           : char;
    aif_area         : integer;
    buff_name        : pathname_type;   { name returned into return buffer}
    buff_offset      : integer;
    buffer           : buffer_type;
    buffer_ptr       : buffer_rec_ptr_type;
    buffer_file_num  : integer;
    buffer_file_name : packed array[1..30] of char;
    continue         : boolean;

    fg_itemnum_array : packed array[1..5] of integer;
    fg_item_array    : packed array[1..5] of globalanyptr;
    fg_item_stat_array:item_status_array_type;
    file_owner       : packed array[1..36] of char;

    file_cnt         : integer;
    filetype         : integer;
    hp_status        : status_type;

    itemnum_array    : packed array [1..5] of integer;
    item_array       : packed array [1..5] of globalanyptr;
    item_status_array: item_status_array_type;
    name_len         : integer;
    num_array_entries: integer;
    overall_status   : status_type;
    recursion        : integer;
    return_array1    : return_array1_type;
    return_array2    : return_array2_type;
    search_key       : max_pathname_type;
    search_path      : pathname_type;       { search path  }
    skip_sw_errs     : boolean;
    sw_overall_status: status_type;
    temp_path        : pathname_type;
    user_id          : integer;
```

```
procedure GETPRIVMODE;   intrinsic;
procedure QUIT;          intrinsic;
procedure HPFOPEN;       intrinsic;

$sysintr 'aifintr.pub.sys'$
procedure AIFACCESSOFF;  intrinsic;
procedure AIFACCESSON;   intrinsic;
procedure AIFSYSWIDEGET; intrinsic;
procedure AIFFILEGGET; intrinsic;


{------------   Print error ---------------------------------------}

procedure ERROR_IN_CALL ( status            : status_type;
                          name              : name_type;
                          item_status_array : item_status_array_type);

var i: integer;
begin
writeln('  ');
writeln('Error in ', name);
writeln('Overall status info = ',status.info, ' subsys= ',status.subsys);
for i := 1 to status.info do
    writeln('Index: ',i,' info= ',item_status_array[i].info,
       ' subsys = ',item_status_array[i].subsys);
end;


begin

{-------------------------------------------------------------------}
{                  Get and validate AIF User ID                     }
{-------------------------------------------------------------------}
GETPRIVMODE;
writeln('Enter a valid user id:');
readln(user_id);
AIFACCESSON ( overall_status, user_id);
if overall_status.all  0 then
   begin
   writeln('AIFACCESSON error. Overall status info = ',overall_status.info,
        ' subsys = ',overall_status.subsys);
   QUIT(997);
   end;
```

```
{-------------------------------------------------------------}
{                 Set up search key pathname                  }
{-------------------------------------------------------------}
search_path.path_str := '/@';
itemnum_array[1]     := 5036;          { pathname item }
item_array[1]        := addr(search_path);

{-------------------------------------------------------------}
{                 Set up file type search criteria            }
{-------------------------------------------------------------}
filetype             := 9;             { Directory object }
itemnum_array[2]     := 5039;          { filetype item }
item_array[2]        := addr(filetype);

{-------------------------------------------------------------}
{  Set up directory recursion level to look at first level      }
{-------------------------------------------------------------}
recursion            := 0;             { Only search first level }
itemnum_array[3]     := 5049;          { Recursion level }
item_array[3]        := addr(recursion);

{-------------------------------------------------------------}
{  Check if user wants to ignore non-fatal errors.  Non-fatal   }
{  errors are those which may prevent a file or directory from  }
{  being opened (e.g. bad ufid, security violation), but they   }
{  won't prevent the rest of the directory from being traversed.  }
{-------------------------------------------------------------}
prompt('Ignore non-fatal errors (Y/N)? ');
readln(answer);
if (answer = 'Y') or (answer = 'y') then
   begin
   skip_sw_errs      := True;
   itemnum_array[4]  := 5050;          { Skip non-fatal errs }
   item_array[4]     := addr(skip_sw_errs);
   itemnum_array[5]  := 0;
   end

else
   itemnum_array[4]     := 0;
aif_area             := 5000;          { File information }
```

```
{-------------------------------------------------------------------------}
{   Open a long mapped file to use as return buffer.  Get ptr to file.   }
{-------------------------------------------------------------------------}
buffer_file_name := '%TEST%';
access           := 4;    { Read/write }
HPFOPEN ( buffer_file_num, hp_status, 2, buffer_file_name, 11, access,
       21, buffer_ptr);
if hp_status.all  0 then
   begin
   writeln('Error during HPFOPEN.  Status.info = ',hp_status.info,
       ' subsys= ',hp_status.subsys);
   QUIT(998);
   end;


{-------------------------------------------------------------------------}
{    Repeat the call to AIFSYSWIDEGET until we have retrieved all files    }
{    that meet the search criteria.                                        }
{-------------------------------------------------------------------------}
search_key        := blanks;
repeat

  {-------------------------------------------------------------------}
  {           Initialize and set up return arrays and buffer          }
  {-------------------------------------------------------------------}
  setstrlen(buffer_ptr^.buff_str, 200000);   { Set length in 1st 4 bytes }
  num_array_entries := 1000;                  { Arrays are 1000 entries}
  item_status_array := init_item_status_array;
  return_array1     := init_return_array1;
  sw_overall_status.all := 0;


  AIFSYSWIDEGET(
                   sw_overall_status,
                   aif_area,            { aif_area = 5000 }
                   return_array1,       { Can return up to 1000 path ids}
                   return_array2,
                   num_array_entries,   { user specified max of 1000 entries}
                   itemnum_array,
                   item_array,
                   item_status_array,
                   search_key,          { defined as max_pathname_type }
                   user_id,
                   buffer_ptr);         { long pointer to a user buffer }
```

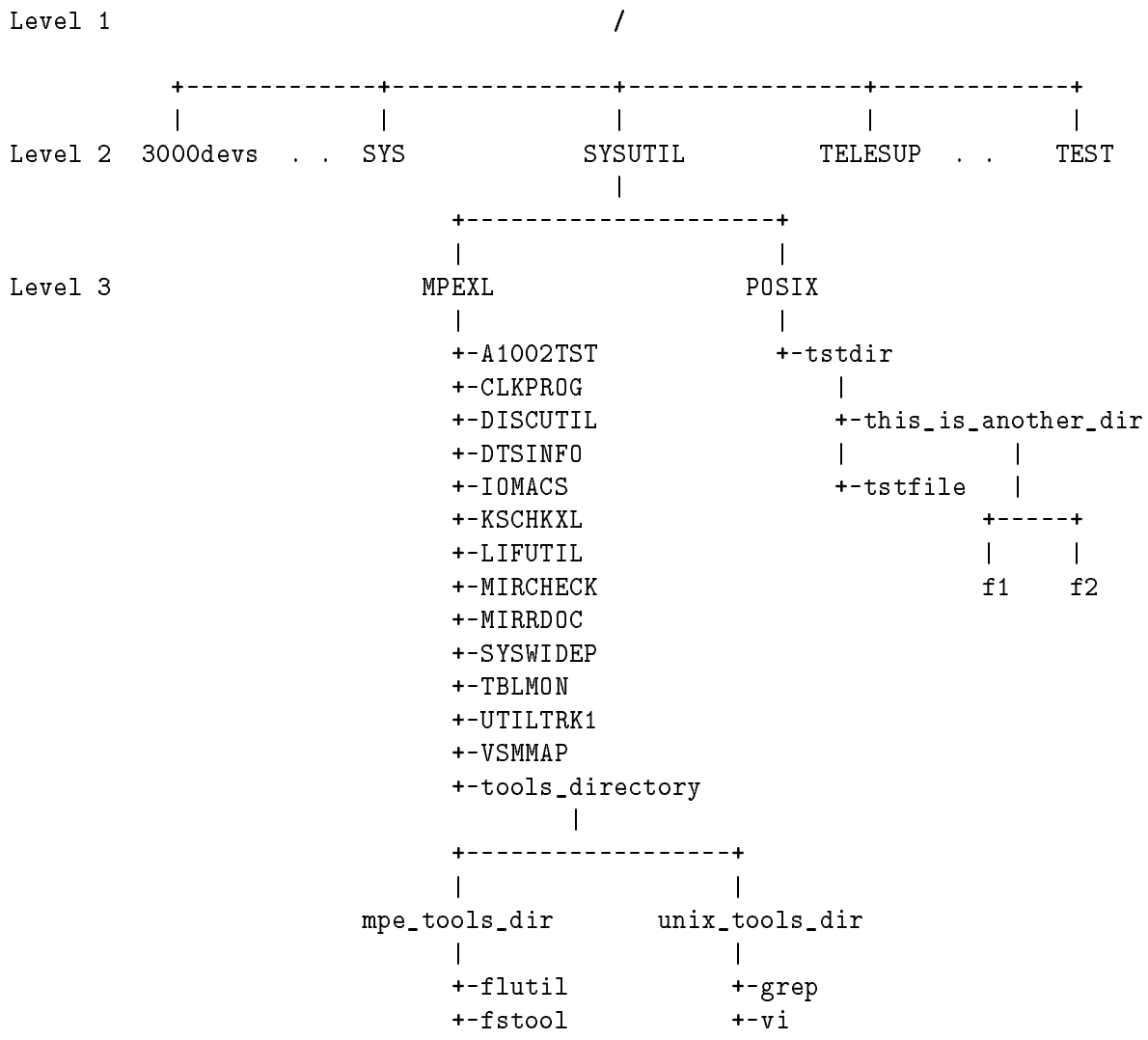
```
{-------------------------------------------------------------}
{ Process error from the AIFSYSWIDEGET call.   If the user did   }
{ not choose to ignore errors, then we can handle it now.  If    }
{ the search key is not equal to blanks, then the error          }
{ was detected on the file returned in search_key, but the       }
{ error won't prevent us from continuing the directory traversal.}
{-------------------------------------------------------------}
if sw_overall_status.all  0 then
   begin
   ERROR_IN_CALL(sw_overall_status, 'AIFSYSWIDEGET', item_status_array);
   continue := false;

   { We only want to continue for some errors, so check the error in the }
   { item status array for the pathname item.                            }
   if (search_key[1]  ' ') then
      begin
      continue := true;
      case item_status_array[1].info of
        -70:  writeln('Directory opened exclusively. Cannot traverse.');
        -72:  writeln('User lacks TD permission on directory component.');
        -75:  writeln('User lacks RD permission on directory component.');
        -83:  writeln('Security violation when traversing directory.');
        -89:  writeln('Error when trying to get flab.  Ufid may be bad!');
        otherwise
              continue := false;
        end;  {case}
      end;

   if continue then
      begin
      temp_path.path_str := '';
      STRMOVE(1024, search_key, 1, temp_path.path_str, 1);
      temp_path.path_str := STRRTRIM(temp_path.path_str);
      writeln('Error occured on file ',temp_path.path_str);
      writeln('Will print files in buffer and continue traversal.');
      writeln('  ');
      end

   else    {  Stop traversal and bail out.  }
      QUIT(999);
   end;    { Endif sw_overall_status.all  0 }
```

```
   {-----------------------------------------------------------------}
   {             Extract pathnames from buffer                       }
   {-----------------------------------------------------------------}
   file_cnt := 1;
   writeln('  ');
   if num_array_entries > 0 then
      writeln('-------------------start of buffer-----------------');

   while file_cnt <= num_array_entries do
      begin

      { Extract return array 2 data }
      name_len    := return_array2[file_cnt].name_len;
      buff_offset := return_array2[file_cnt].buff_offset;
      buff_name.pathname   := blanks;
      buff_name.path_str   := '';      { Initialize string }
      strmove( name_len, buffer_ptr^.buff_str, ((buff_offset-4)+1),
            buff_name.path_str, 1);
      writeln(' Pathname is ',buff_name.path_str);

      { Call AIFFILEGGET to get the file owner for each file in buff_name }
      fg_itemnum_array[1] := 5041;         { File owner }
      fg_item_array[1]    := addr(file_owner);
      fg_item_stat_array  := init_item_status_array;
      fg_itemnum_array[2] := 0;            { end the item list }

      AIFFILEGGET(overall_status,
                  fg_itemnum_array,
                  fg_item_array,
                  fg_item_stat_array,
                  {ufid},
                  {filename},
                  {tempfile},
                  user_id,
                  {pathid},
                  buff_name);
      if overall_status.all = 0 then
         writeln(' File owner is ',file_owner)
      else
         ERROR_IN_CALL(overall_status, 'AIFFILEGGET', fg_item_stat_array);
      writeln;
      file_cnt := file_cnt + 1;

      end;    { end do while file_cnt <= num_array_entries }

   if num_array_entries > 0 then
      writeln('-------------------end of buffer-----------------');
until (search_key[1]=' ');

end.    { end program  }
```

Example 6 - HFS directory traversal

Directory Traversal Examples

The following examples illustrate various pathnames you can specify as input to `AIFSYSWIDEGET` and the format of the names you will receive as output from `AIFSYSWIDEGET`. The examples assume the following directory structure.



Absolute Pathnames With Recursion

The following three examples illustrate the absolute pathnames that would be returned by AIFSYSWIDEGET if an absolute pathname was specified as the fileset for item 5036. Assume that SYSWIDEP is a program which calls AIFSYSWIDEGET and passes in the recursion level input by the user as item 5041.

```
:CHGROUP MPEXL
:RUN SYSWIDEP

Path? >/SYSUTIL/MPEXL/@
Recursion level? >99

/SYSUTIL/MPEXL/A1002TST
/SYSUTIL/MPEXL/CLKPROG
/SYSUTIL/MPEXL/DISCUTIL
/SYSUTIL/MPEXL/DTSINFO
/SYSUTIL/MPEXL/IOMACS
/SYSUTIL/MPEXL/KSCHKXL
/SYSUTIL/MPEXL/LIFUTIL
/SYSUTIL/MPEXL/MIRCHECK
/SYSUTIL/MPEXL/MIRRDOC
/SYSUTIL/MPEXL/SYSWIDEP
/SYSUTIL/MPEXL/TBLMON
/SYSUTIL/MPEXL/UTILTRK1
/SYSUTIL/MPEXL/VSMMAP
/SYSUTIL/MPEXL/tools_directory
/SYSUTIL/MPEXL/tools_directory/.
/SYSUTIL/MPEXL/tools_directory/..
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/.
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/..
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/flutil
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/fstool
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/.
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/..
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/grep
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/vi
END OF PROGRAM
```

```
:RUN SYSWIDEP

Path? >/SYSUTIL/@/@dir@
Recursion level? >99
/SYSUTIL/MPEXL/tools_directory
/SYSUTIL/MPEXL/tools_directory/.
/SYSUTIL/MPEXL/tools_directory/..
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/.
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/..
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/flutil
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/fstool
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/.
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/..
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/grep
/SYSUTIL/MPEXL/tools_directory/unix_tools_dir/vi
/SYSUTIL/POSIX/tstdir
/SYSUTIL/POSIX/tstdir/.
/SYSUTIL/POSIX/tstdir/..
/SYSUTIL/POSIX/tstdir/this_is_another_dir
/SYSUTIL/POSIX/tstdir/this_is_another_dir/.
/SYSUTIL/POSIX/tstdir/this_is_another_dir/..
/SYSUTIL/POSIX/tstdir/this_is_another_dir/f1
/SYSUTIL/POSIX/tstdir/this_is_another_dir/f2
/SYSUTIL/POSIX/tstdir/tstfile
END OF PROGRAM


:RUN SYSWIDEP

Path? >/SYSUTIL/@/@dir@/@/f@
Recursion level? >99

/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/flutil
/SYSUTIL/MPEXL/tools_directory/mpe_tools_dir/fstool
/SYSUTIL/POSIX/tstdir/this_is_another_dir/f1
/SYSUTIL/POSIX/tstdir/this_is_another_dir/f2
END OF PROGRAM
```

Pathnames Relative to CWD With No Recursion

The following three examples illustrate the relative pathnames that would be returned by AIFSYSWIDEGET if a relative pathname was specified as the fileset for item 5036.

```
:RUN SYSWIDEP

Path? >./@
Recursion level? >0

./A1002TST
./CLKPROG
./DISCUTIL
./DTSINFO
./IOMACS
./KSCHKXL
./LIFUTIL
./MIRCHECK
./MIRRDOC
./SYSWIDEP
./TBLMON
./UTILTRK1
./VSMMAP
./tools_directory
END OF PROGRAM

:CHDIR /SYSUTIL/MPEXL/tools_directory

:RUN SYSWIDEP.MPEXL

Path? >./@
Recursion level? >0

./.
./..
./mpe_tools_dir
./unix_tools_dir
END OF PROGRAM
```

```
:RUN SYSWIDEP.MPEXL

Path? >./@
Recursion level? >99

./.
./.
./..
./mpe_tools_dir
./mpe_tools_dir/.
./mpe_tools_dir/..
./mpe_tools_dir/flutil
./mpe_tools_dir/fstool
./unix_tools_dir
./unix_tools_dir/.
./unix_tools_dir/..
./unix_tools_dir/grep
./unix_tools_dir/vi
END OF PROGRAM
```

What Are '.' and '..' Files?

File names represented by a dot (.) or two dots (..) are directory files which in the above example represent the current directory and the parent of the current directory respectively. The current directory or the parent directory may refer to an HFS directory file or to an MPE directory node (for example, $FILESET_NODE, $GROUP_NODE).

For example, using the directory structure shown above, the following files refer to MPE directory structures:

```
/SYSUTIL/MPEXL/.                    <- refers to $FILESET_NODE.MPEXL.SYSUTIL
/SYSUTIL/MPEXL/tools_directory/..   <- refers to $FILESET_NODE.MPEXL.SYSUTIL
```

These files are not shown by a `LISTFILE` command, but they are returned by `AIFSYSWIDEGET`.

Example 7 - Using Magneto-Optical AIFs

Below is a sample program illustrating the usage of the Magneto-Optical AIFs. It allocates the first available drive that can access the media labeled MYMEDIA, mounts the media MYMEDIA on this drive, accesses files on the media and then dismounts the media and deallocates the drive.

```
$standard_level 'ext_modcal'$
Program MO_Sample (input, output);

Type
  media_label_type = record
                       media_name : packed array [1..32] of char;
                       subname1   : packed array [1..16] of char;
                       subname2   : packed array [1..16] of char;
                     end;

  name_type = packed array [1..16] of char;

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

  item_status_array_type = array [1..4] of status_type;

Var
  drive_ldev        : integer;
  itemnum_array     : packed array [1..4] of integer;
  item_array        : packed array [1..4] of globalanyptr;
  item_status_array : item_status_array_type;
  media_id          : media_label_type;
  overall_status    : status_type;
  user_id           : integer;
  volume_set        : packed array[1..8] of char;

procedure GETPRIVMODE;      intrinsic;
procedure QUIT;             intrinsic;

$sysintr 'aifintr.pub.sys'$
procedure AIFMOALLOCATE;    intrinsic;
procedure AIFMODEALLOCATE;  intrinsic;
procedure AIFMOMOUNT;       intrinsic;
procedure AIFMODISMOUNT;    intrinsic;
```

```
{------------------------ Print error and bail out --------------------}

procedure ERROR_IN_CALL ( status            : status_type;
                          name              : name_type;
                          item_status_array : item_status_array_type);
Var
  local_status : status_type;

begin
  writeln('Error in ', name);
  if (status.all < 0) then
    writeln('Overall status info = ', status.info, ' subsys= ', status.subsys)
  else
    begin
      local_status.all := item_status_array[status.info].all;
      writeln('Item #: ', status.info:1, ' status info = ',
              local_status.info:1, ' subsys = ', local_status.subsys:1);
    end;
  QUIT(999);
end;


begin

{---------------------------- Get AIF user ID -------------------------}
GETPRIVMODE;
writeln('Enter a valid user id:');
readln(user_id);


{----- Allocate a drive that can access the media labeled MYMEDIA -----}
itemnum_array[1] := 17103;              { Allocate by media name }
media_id.media_name := 'MYMEDIA';
media_id.subname1 := '@';               { Ignore subname1 }
media_id.subname2 := '@';               { Ignore subname2 }
item_array[1] := addr(media_id);
item_status_array[1].all := 0;
itemnum_array[2] := 0;

AIFMOALLOCATE (overall_status,
               drive_ldev,     { The allocated drive is returned in drive_ldev }
               itemnum_array,
               item_array,
               item_status_array,
               user_id);

if overall_status.all  0 then
  ERROR_IN_CALL(overall_status, 'AIFMOALLOCATE', item_status_array);
```

```
{------- Mount the media labeled MYMEDIA in the allocated drive -------}
{------- and return the volume set name for this media. ---------------}
itemnum_array[1] := 17303;              { Return volume set name }
item_array[1] := addr(volume_set);
item_status_array[1].all := 0;
itemnum_array[2] := 0;

AIFMOMOUNT (overall_status,
            drive_ldev,
            media_id,
            itemnum_array,
            item_array,
            item_status_array,
            user_id);

if overall_status.all  0 then
  ERROR_IN_CALL(overall_status, 'AIFMOMOUNT', item_status_array);

{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}
{  At this point, the media (volume set) is mounted and can be used    }
{  like any other user volume set.  Groups and accounts can be created }
{  programmatically or through commands (for example, NEWGROUP ;ONVS=) }
{  and files can also be created on the media (for example, HPFOPEN,   }
{  BUILD).                                                             }
{-------------------------------------------------------------------------}
{-------------------------------------------------------------------------}

{----------------------- Dismount the media ----------------------------}
AIFMODISMOUNT (overall_status,
               drive_ldev,
               ,
               ,
               ,
               user_id);
if overall_status.all  0 then
  ERROR_IN_CALL(overall_status, 'AIFMODISMOUNT', item_status_array);

{--------------------- Deallocate the drive --------------------------}
AIFMODEALLOCATE (overall_status,
                 drive_ldev,
                 ,
                 ,
                 ,
                 user_id);
if overall_status.all  0 then
  ERROR_IN_CALL(overall_status, 'AIFMODEALLOCATE', item_status_array);

end.
```

D

Glossary

Absolute Pathname — A pathname that begins with the root directory, such as /SYS/PUB/TDP. See also *pathname* and *relative pathname*.

Artificial Member — Workgroup membership is composed of natural and artificial members. A process becomes an artificial member when it is explicitly placed into the workgroup via :ALTPROC or AIFPROCPUT. A process remains an artificial member of its assigned workgroup until:

- the workgroup is purged, or
- the process is explicitly released from its artificial assignment via the :ALTPROC command or AIFPROCPUT.

That is, an artificial member is not affected by changing one of the process attributes used in workgroup assignment. In addition, a scan would only effect the process if the process' workgroup had a purge pending.

Asynchronous Port — A port that provides the capability of interrupting the creator upon receipt of a message to that port. User code will only be interrupted when it is executing at privileged levels 2 or 3, and it is not set critical or in system code. Asynchronous ports may only have one receiver and it must be the port creator. Asynchronous ports are NOT permanent.

Base — The base is the highest priority value (lowest numeric value) of processes within that workgroup (BASE=Value). Values can range between the priority values of 150 and 255. Internally the priorities range from 32767 to 0. AIFSCGET/PUT return or modify internal priorities. Processes will begin their transactions at the base priority and decay as they consume CPU. The base is a required workgroup characteristic.

Boost Property — The boost property can be set to either decay or oscillate (BOOST={ DECAY,OSCILLATE}). A value of decay is the default and means the priority of a process within that workgroup will begin at the base and decay as the process consumes CPU. A value of oscillate indicates that the priority of the process will be reset to the base if it decays to the limit (the priority of the process will oscillate between the base and limit priorities). The boost property is an optional workgroup characteristic with a default value of DECAY.

CI — CI is an abbreviation for the command interpreter. The CI analyzes and processes commands entered during a session or submitted as part of a job.

Connectionless Send — The ability to send to an AIF port without having previously done an `AIFPORTOPENi` on that port. The only type of Vconnectionless send that can be done is a "no wait" send.

Constant Priority Process — This process does not decay and remains at the same level regardless of the queue the process is in. By default, only the processes in AS, BS, and certain processes in CS queue of type UCOP and System belong to this process category.

CM Files — The MPE/iX file system currently consists partially of NM files and CM files. Consequently, CM code handles certain types of files after switching to CM. Files that require switching to CM are called CM files.

Current Working Directory (CWD) — The directory in which you are working and from which relative pathnames are resolved. See also *directory* and *relative pathname*.

CWD — An acronym for Current Working Directory.

Decayable Boosting — This causes the priority to descend gradually through a series of drops until a transaction is completed or decays to the bottom of a queue. After this occurs the priority resets to the base of the CS queue.

Default Workgroups — One of five system-defined workgroups created to provide backward compatibility with the five scheduling queues. These workgroups, AS_Default, BS_Default, CS_Default, DS_Default, and ES_Default, are created with the same scheduling

characteristics as their namesake and have the scheduling queue as their only membership criterion. The characteristics of the AS_Default and BS_Default workgroups cannot be changed. The characteristics of the CS_Default, DS_Default, and ES_Default can be changed through AIFSCPUT and AIFWGPUT.

directory A special kind of file that contains entries that point to other files. It acts like a container for files and other directories. On MPE/iX, accounts and groups are special types of directories.

Envelope The envelope in the context of IPC and message-passing is analogous to the envelope you use to send a letter through the Postal Service. The envelope is the *overhead* portion of the total data required to send a message from a *sender* process to a *receiver* process. The envelope contains the priority of the message, reply information (the return address), and other miscellaneous information needed for routing and scheduling.

Envelope Code The envelope code is an integer value that can be passed with any message. This value is available to a *receiver* process without reading the actual message. If the message sent can fit within an integer value, a zero length message can be sent with the actual message contained entirely in the envelope code.

FIFO FIFO is an abbreviation for first in first out. Since messages can be assigned a priority between 0 and 31, where 0 has the highest priority, a message of priority 0 is received before a message of priority 1, even if the priority 1 message arrived in the port first. Messages sent with the same priority are delivered in the same order they were sent, or FIFO.

File Code A file code is a four-digit integer that identifies the function of a special purpose file. For example, a V/3000 forms file has a file code of 1035. For a list of file code numbers and their meanings, consult the *File System Reference Manual*.

File Equation A file equation directs the input to, or output from, a program, job, or session. You create a

	file equation by using the File command to equate a file name to another file or device.
File Name	Most of the AIF interfaces accept and return fully qualified file names in a single standard format.
File Number	Each OPEN (FOPEN or HPFOPEN) returns a number to the caller which is a process-specific handle for this instance of the OPEN. For the remainder of the section, a file number always signifies the file context addressed by this process-specific handle.
Handler	The code that is specified to handle interrupts from an asynchronous port. This user-written routine will be required to have only one parameter, an AIF port ID. This ID is passed to the handler upon its invocation, by the AIF subsystem. The address of the handler must be provided at creation time to the AIFPORTOPEN for an asynchronous port.
HFS	An acronym for the hierarchical file system.
hierarchical file system (HFS)	A file system that is tree structured and can contain files at many different levels. This file organization is obtained through the use of directories, which can contain files and other directories.
Home Group	The home group is the group assigned to a user when the user name is defined with the Newuser command. The group is the user's default logon group if a group name is not specified with the Hello or Job command.
IPC	IPC is an abbreviation for Inter Process Communication, which is message-passing between processes. Although normally occurring between two or more different processes, the communication can also occur between a single process and itself.
Job	A job is a sequence of instructions issued to the computer that does not require an interactive dialog between the user and the computer. Each job on the system is uniquely identified by a job number.
Job/Session Number	A job/session number uniquely identifies either a job or session.

Job State
A generic term used for the stages that a job or session might pass through during its lifespan.

Limit
The limit is the lowest priority (highest numeric value) of processes within that workgroup (LIMIT=value). Values can range between the priority values of 150 and 255. Internally the priorities range from 32767 to 0. AIFSCGET/PUT return or modify internal priorities. Process priorities within the workgroup will not decay beyond the limit. If the boost property for the workgroup is oscillate, process priorities will be reset to the base value once they decay to the limit. The limit is a required workgroup characteristic.

Linked Spoolfile
A linked spoolfile has an entry in the SPFDIR and resides in the HPSPOOL account. Input spoolfiles reside in @.IN.HPSPOOL. Output spoolfiles reside in @.OUT.HPSPOOL. If a user copies a spoolfile from OUT.HPSPOOL to his or her local group and account, the copy has no entry in the SPFDIR and is therefore not a linked spoolfile. Refer to the spooler management routine AIFSpoolfLink for further information.

Logon
Logon is the job/session, user and account name associated with a process. The logon of a process can change dynamically through AIFCHANGELOGON. Logon is one of the process attributes used to determine workgroup membership. Therefore, a change in logon may result in an immediate change in workgroup assignment.

Mail Slot
The front panel storage slot used to insert or remove Magneto-Optical Media in an optical disk library system.

Maximum CPU Percentage
The maximum CPU percentage is a upper bound for the amount of CPU the processes in a workgroup can consume relative to other workgroups. The maximum CPU percentage value can be used to limit the amount of CPU consumed by a workgroup. This control may result in the system idling if the workgroup hits its maximum percentage and there are no other users who want the CPU. The default value is 100%.

Maximum Quantum The maximum quantum is an upper bound for the dynamically calculated quantum (average transaction time) value for that workgroup (MAXQUANT=Value). Values range between 0 and 32767. The maximum quantum is an optional workgroup characteristic with a default value of 1000.

Media Label A record defining the label for Magneto-Optical Media which consists of three parts including media_name, subname1, and subname2.

Media Name A packed array of 1 to 32 characters used to identify the first part of the media label

Media Slot A number specifying a Magneto-Optical disk library system media storage slot.

Membership Criteria The membership criteria of a workgroup is composed of a number of category specifications. Three categories are currently supported (*logon, program, and scheduling queue*). For a given workgroup, at least one category must be specified. If multiple specifications are specified, a process must match one specification from each category. Categories not specified take their default values.

The following table lists the default values for the membership criteria:

Table D-1.
Membership Criteria Default Values

Membership Criteria	Default Values
Logon	@,@.@ (any jobname,user.account)
Program	@.@.@ (any program)
Scheduling Queue	AS, BS, CS, DS, ES (any queue)

Message The Message is the portion of the overall package handled internally by the AIF Ports code that is delivered to the *receiver* process. It is the data that is "sent."

Message File A message file acts as a first-in-first-out queue of records, with entries made by FWRITE and deletions made by FREAD. These are often used for interprocess communication

by having one process submit records while another process removes them.

Minimum CPU Percentage

The minimum CPU percentage is a lower bound for the amount of CPU the processes in a workgroup can consume relative to other workgroups. The minimum CPU percentage value can be used to *guarantee* a certain amount of CPU to a workgroup. Note that the CPU consumption of the workgroup may not precisely match the specified minimum CPU percentage if there is insufficient demand within the workgroup. The default value is 0%.

Minimum Quantum

The minimum quantum is a lower bound for the dynamically calculated quantum (average transaction time) value for that workgroup (MINQUANT=Value). Values range between 0 and 32767. The mimimum quantum is an optional workgroup characteristic with a default value of 1 for user workgroups and CS_Default workgroup. The default value for DS_Default and ES_Default is 2000.

MPE file

The term MPE file refers to a file that can be represented using MPE semantics (for example, CI.PUB.SYS).

Natural Member

A process becomes a natural member of a workgroup when it is placed into the workgroup via the system. The system will scan the ordered list of workgroups, selecting the first workgroup whose membership criteria match the process' attribute.

NM Files

The MPE/iX file system currently consists partially of NM files and CM files. Consequently, CM code handles certain types of files after switching to CM. Files that do not require switching to CM are called NM files.

NMS

NMS is an abbreviation for the Native Mode Spooler. This new MPE/iX native mode spooler replaces the previous CM SPOOLER and SPOOK.

pathname

A pathname specifies where a particular file or directory is within the directory structure; that is what path the system must take when traversing the directory. See also absolute pathname and relative pathname.

PID

PID is an abbreviation for Process ID. Just as every process is assigned a PIN #, in MPE/iX every process is also assigned a PID. The PID is a 64-bit long integer comprised of the machine #, the PIN #, and a reuse counter.

PIN

PIN is an abbreviation for Process Identification Number. In MPE/iX every process is assigned a PIN #. The PIN is a 16-bit short integer.

Port

A Port refers to the collection of data structures managed by the AIF Ports procedures. It provides a level of abstraction when sending a message form one process to another. The sending process does not need to be explicitly aware of exactly which process reads the message; it simply sends a message to a given Port knowing that some process will eventually read it.

Likewise, the *receiver* does not necessarily need to know which process sent it a message; the *receiver* only needs to know that the message came from a given Port. A message is considered to pass through a Port during the send/receive cycle.

Port Manager

As part of the feature set provided with the AIF Ports facility, a process can be designated as the *Port Manager* for the Port. When a *Port Manager* is not involved in the message transfer, the AIF Ports code prioritizes a message from the *sender* along with any previously sent but unreceived messages, then signals a *receiver* process ready for another message that a message is available.

When a *Port Manager* process is involved, the AIF Ports code simply stores the message from the *sender* in the Port data structures, then sends the envelope portion of the message to the *Port Manager* so it can assume processing of the message.

Port Name

A Port name is a name given to a Port, which can consist of from 1 to 16 characters and can contain any characters. The name is upshifted before use.

Port Password

A Port password is a password to be associated with a Port. The process that

creates a Port establishes the Port password. All subsequent opens must use the same password.

POSIX Portable Operating System Interface. A set of standards that address various areas of operating system technology. The POSIX standards describe functions of an operating system interface that applications use to become POSIX-compliant. The main point of POSIX is to facilitate software portability and minimize porting costs.

Priority Boost A priority boost raises the priority of a process. This occurs at the end of a transaction when a process holds a resource that a higher priority process needs, when a process has a stalled transaction, or when Break or Ctrl-Y must be processed.

Private Spoolfile A private spoolfile is HPFOPENed with the PRIVATE option. Under NMS all input spoolfiles are private spoolfiles, and a user can designate output spoolfiles private for security purposes. Refer to the NMS manual for further information on private spoolfiles.

Process-specific File The same physical file maybe opened more than once by the same process. Some of the file context is common for all the OPENs issued by a process against a physical file. This kind of information is referred to as process-specific information.

Program File The name of the program file that is currently loaded for execution by the process. This name may change if the process makes use of the POSIX exec system call. The program file is one of the process attributes used to determine the workgroup membership. Therefore, changing the program file may result in an immediate change in workgroup assignment.

Receiver Process The receiver process is the complement to a *sender process*. If one process is sends messages, the *receiver process* reads these messages.

Record Pointer The file system maintains information about where the user is located in the file (what the next read fetches and where the next write is dispatched). The record pointer, the record number, and the offset within the record all

provide the complete context. When a record pointer is shared, all three are shared.

Relative Pathname A pathname that is interpreted from the current working directory. For example, ./dir1/longfilename refers to the file longfilename in directory dir1 in the current working directory.

Return_array The system-wide interface returns values in arrays of this type. The actual structure of the array varies depending on the type of keys passed, but the general form is: *Array [1..x] of appropriate type*. *x* represents any integer and *appropriate type* is specified in AIFSysWideGet.

Scheduling Characteristics The scheduling characteristics of the workgroup determine the scheduling policies which govern the processes within that workgroup. The **base** and **limit** priorities determine the range of the priority values for processes within the workgroup, while the **quantum bounds** define the range over which the quantum can change. The **timeslice** and **boost property** values also determine the scheduling behavior. The other scheduling characteristic is **CPU Percentage bounds**.

The following table lists the default values for the scheduling characteristics:

Table D-2. Scheduling Characteristics Default Values

Scheduling Characteristics	Default Values
Boost Property	Decay
Minimum CPU Percentage	0%
Minimum Quantum	1 (2000 for DS_Default and ES_Default)
Maximum CPU Percentage	100%
Maximum Quantum	2000
Timeslice	200 (1000 for AS_Default and BS_Default)

Scheduling Queue In the current implementation, scheduling queue can mean one of two things. First, a scheduling queue is a process attribute that can be set by the user (e.g., :RUN foo;PRI=BS). This attribute can also be changed dynamically through :ALTPROC , the intrinsic GETPRIORITY, or the AIF,

AIFPROCPUT. Second, scheduling queue refers to a collection of processes with similar scheduling characteristics. MPE/iX currently supports five queues. The AS and BS queues are typically used for system processes, the CS queue is typically used for interactive users, while the DS and ES queues are typically used for batch jobs.

Scheduling queue, as a process attribute, is an integral part of the workgroup concept. This is one of the process attributes used to determine workgroup membership. Because of its dynamic nature, a change in the process' queue attribute results in an immediate change in the process' workgroup assignment.

Search_key The system-wide interface returns arrays of keys. The number of keys returned in a call depends on the space that you allocate. If more keys can be returned then this is indicated in the status, a special key is returned that can be used in a subsequent call to *AIFSysWideGet* to start the scan from that search key without repeating the keys returned before. The search key should be defined as *array [1..48] of char*.

Sender Process The *sender process*, sometimes referred to as the *sender*, enables a message to be sent and received by another process.

Session A session is an interactive dialog between the user and the computer. Each session on the system is uniquely identified by a session number.

SPFDIR This is an abbreviation for spoolfile directory, which is the table that the NMS uses to maintain information about spoolfiles.

SPIT This is an abbreviation for the Spooling Process Information Table, which is the table that NMS uses to maintain information about spooling processes.

Spoolfile The spoolfiles generated by the file system for the NMS are ordinary disc files. This prevents input and output spoolfiles from being lost during system boots as they currently are. A new file type identifies the files as spoolfiles and allows them to be managed in this manner.

Two new file codes have also been assigned: 1515 for input spoolfiles and 1516 for output spoolfiles. Input spoolfiles are created in the IN group of the reserved account HPSPOOL, and output spoolfiles are created in the OUT group of the HPSPOOL account.

Streams LDEV The streams LDEV is the device specified with the Streams command to be used as the input device for all jobs on the system. This device should not actually exist, as it is a 'pseudo-device' that must be configured with the device class JOBTAPE.

Subname1 A packed array of 1 to 16 characters used to identify the second part of the media label.

Subname2 A packed array of 1 to 16 characters used to identify the third part of the media label.

Surface A number of either 0 or 1. The number zero specifies the "A" side of a Magneto-Optical Media and the number one specifies the "B" side. Sometimes referred to as "side".

System Average Quantum (SAQ) System Average Quantum determines how rapidly process priorities decay. There are different SAQs for the CS, DS and ES queues. Within the CS queue, the SAQ is adjusted as processes complete transactions and represents the average transaction time of processes in the CS queue. The SAQ for the DS and the ES queues is a user-configurable value chosen to represent the average transaction time of these queues.

System Logging System logging is a facility that records the occurrence of specific events and system resource usage into the system log files on a job/session basis. The system manager can enable or disable system logging types.

System Process A system process is one that is a child of PROGEN or has inherited system process status from its parent. By definition, a system process executes with a non-decayable (linear) priority. However, a process does not need to be a system process to have a non-decayable priority. Process Management considers a system process an integral part of the OS and will abort the system if a system process dies.

Timeslice

The timeslice is the maximum number of milliseconds a process in that workgroup can hold the CPU before returning to the Scheduler to have its priority recalculated(TIMESLICE=value). Values must be multiples of 100, with a minimum value of 100 and a maximum value of 32700. The timeslice is an optional workgroup characteristic with a default value 200 milliseconds for CS_Default, DS_Default, ES_Default and user-defined workgroups. The default value for AS_Default and BS_Default workgroups is 1000.

Transaction

A transaction is comprised of a series of events. Most commonly, a transaction is the action performed between terminal read waits. A transaction is also considered complete when a process pauses at length (more than two seconds), blocks on a call to IO_Wait, or blocks waiting for IPC.

UFID

UFID is an abbreviation for for Unique File Identifier. It is a unique name to a single file throughout the life of a system. It is unique even across system boots.

User Files

The files opened for a particular user could have been opened either by an explicit OPEN (FOPEN or HPFOPEN) by the user program, or by the system on behalf of the user. The files opened explicitly are called user files and the remainder are non-user files. This procedure can be used to modify the information about user files only.

Workgroup

A workgroup is a collection of processes with identical scheduling characteristics. The scheduling characteristics include base and limit priority, timeslice value, boost property, etc.

Workgroup membership is determined by matching specific process attributes against a set of predefined membership criteria. The process attributes selected include logon, program file, profile, and scheduling queue.

Workgroup membership criteria and scheduling characteristics are determined by the System Manager.

Index

B

C

E

G

N

P

S