Guardian Procedure Errors and Messages Manual

Abstract

This manual describes the Guardian messages for HP systems that use the HP NonStop™ operating system. The manual covers the following types of messages: error codes and error lists associated with Guardian procedure calls, the interprocess messages sent to application programs by the operating system and the command interpreter, traps, and signals.

Product Version

N/A

Supported Release Version Updates (RVUs)

This publication supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, and G06.15 and all subsequent G-series RVUs, until otherwise indicated by its replacement publications. Additionally, all considerations for H-series throughout this manual will hold true for J-series also, unless mentioned otherwise.

Part Number	Published
522628-010	August 2010

Document History

Part Number	Product Version	Published
522628-005	N/A	July 2005
522628-006	N/A	August 2007
522628-008	N/A	May 2008
522628-009	N/A	February 2010
522628-010	N/A	August 2010

Legal Notices

Guardian Procedure Errors and Messages Manual

Index Figures Tables

2. File-System Errors (continued)

Figures

Tables

Tables (continued)

What’s New in This Manual

Manual Information

Abstract

This manual describes the Guardian messages for HP systems that use the HP NonStop^{TM} operating system. The manual covers the following types of messages: error codes and error lists associated with Guardian procedure calls, the interprocess messages sent to application programs by the operating system and the command interpreter, traps, and signals.

Product Version

N/A

Supported Release Version Updates (RVUs)

This publication supports J06.03 and all subsequent J-series RVUs, H06.03 and all subsequent H-series RVUs, and G06.15 and all subsequent G-series RVUs, until otherwise indicated by its replacement publications. Additionally, all considerations for H-series throughout this manual will hold true for J-series also, unless mentioned otherwise.

Part Number	Published
522628-010	August 2010

Document History

Part Number	Product Version	Published
522628-005	N/A	July 2005
522628-006	N/A	August 2007
522628-008	N/A	May 2008
522628-009	N/A	February 2010
522628-010	N/A	August 2010

New and Changed Information

Changes to the H06.21/J06.10 manual:

- Added a description for error 35 on page 2-16.
- Added file-system errors 758 through 766 on page 2-84.
- Updated Note under Signals and Trap Numbers on page 21-2.
- Updated the description of error code 72 on page 6-10.

- Removed the instance of "-instance_data data2protected" from recovery action 10 of error code 77 on page 6-28.

Changes to the H06.20/J06.09 Manual

- Added the following new error code descriptions:
    - Errors 700 on page 2-76 to 711 on page 2-79.
    - Error 734 on page 2-83.
    - Error 735 on page 2-84.
    - Error 899 on page 2-86.

Changes to the H06.14/J06.03 Manual

- Supported release statements have been updated to include J-series RVUs.

Changes to the H06.11 Manual

- Updated the name of error 733 (%1335) FEBRANCHISFAILED on page 2-83.
- Added these new errors:
    - 3502 (%6656) FEREQUESTALLOCATIONFAILURE on page 2-87
    - 538 SIOERR^EXTSIZE^OVERFLOW: Extent size is greater than 65535 pages on page 3-7
    - 539 SIOERR^PAGEWRITE^OVERFLOW: The highest possible page in EDIT file has been written. on page 3-8
    - 2081 (%004041) Bad internal format. on page 4-9
- Added subcode 75 on page 6-10 in the Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3xx table.

Changes to the G06.24 Manual

- Unless otherwise indicated in the text, discussions of native mode behavior, processes, and so forth apply to both the TNS/R code that runs on G-series systems and to the TNS/E code that runs on H-series systems
    - New error messages 80, 81, 82,83, 84, and 99 are added to Section 6, Process Creation Errors.
    - Table 6-2, Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3xx, on page 6-7 describes new error subcodes 72-79.
    - New details for Errors 71, 77, and 78 are added to Section 6, Process Creation Errors.

- New Cause and Recovery details for Error 77 are added to Section 6, Process Creation Errors.
- New Cause and Recovery details for Error 71 are added to Section 6, Process Creation Errors.

About This Manual

Purpose of This Manual

This manual describes the Guardian messages for HP systems that use the HP NonStop operating system. It covers the following types of messages:

- The error codes and error lists associated with Guardian procedure calls
- The interprocess messages sent to application programs by the operating system and the command interpreter
- The trap numbers, signals, and the error list for traps and signals

Each message description explains the cause of the message, states the effect on the system, and provides a recommended recovery action.

Who Should Read This Manual

This manual is intended mainly for application programmers whose programs call Guardian procedures directly. Applications can check for errors returned by these procedures, and subsequent action can be initiated based on the error returned.

Other users who can benefit from this manual include interactive users of HP NonStop systems, system managers, and system operators.

How This Manual Is Organized

This manual has the following organization.

- Section 1, Introduction, provides an overview of the various messages described in this manual.
- Section 2, File-System Errors, describes the file-system error codes and error lists.
- Section 3, Sequential I/O Errors, describes the error codes produced by the sequential I/O (SIO) procedures.
- Section 4, DEFINE Errors, describes the errors that relate specifically to DEFINE attribute sets.
- Section 5, NEWPROCESS AND NEWPROCESSNOWAIT Errors, describes the error codes and error lists produced by the process-control procedures NEWPROCESS and NEWPROCESSNOWAIT.
- Section 6, Process Creation Errors, describes the error codes and error lists produced by the process-creation procedures PROCESS_LAUNCH_ and PROCESS_CREATE_. It also describes Guardian error codes returned by the PROCESS_SPAWN_ procedure. Open System Services (OSS) error codes and

the error list returned by the PROCESS_SPAWN_ procedure are listed in Section 9, PROCESS_SPAWN_ Open System Services (OSS) Errors.

- Section 7, PROCESS_GETINFOLIST_ Errors, describes the error codes produced by the PROCESS_GETINFOLIST_ procedure.
- Section 8, PROCESS_GETPAIRINFO_ Errors, describes the error codes produced by the PROCESS_GETPAIRINFO_ procedure.
- Section 9, PROCESS_SPAWN_ Open System Services (OSS) Errors, describes the OSS error codes and error list produced by the process-creation procedure PROCESS_SPAWN_. Guardian error codes returned by both the PROCESS_SPAWN_ and PROCESS_CREATE_ procedures are documented in Section 6, Process Creation Errors.
- Section 10, ALLOCATESEGMENT Errors, describes the error codes and error lists produced by the ALLOCATESEGMENT procedure.
- Section 11, SEGMENT_ALLOCATE_ Errors, describes the error codes and error lists produced by the SEGMENT_ALLOCATE_ procedure.
- Section 12, USESEGMENT Errors, describes the error list associated with the USESEGMENT procedure.
- Section 13, SEGMENT_USE_ Errors, describes the error list associated with the SEGMENT_USE_ procedure.
- Section 14, Subsystem Programmatic Interface (SPI) Errors, describes the error codes issued by the SPI procedures and the error lists associated with those procedures.
- Section 15, EDITREAD and EDITREADINIT Errors, describes the errors returned by the EDITREAD and EDITREADINIT procedures.
- Section 16, IOEdit Errors, describes the error messages returned by the IOEdit procedures. These errors include file-system errors that have specific meaning for IOEdit.
- Section 17, Formatter Errors, describes the error codes produced by the FORMATDATA[X] procedures.
- Section 18, INITIALIZER Errors, describes the error messages produced by the INITIALIZER procedure.
- Section 19, Interprocess Command Interpreter Messages, describes the command interpreter system messages that an application can receive through its $RECEIVE file.
- Section 20, System Messages, describes the operating-system messages that an application can receive through its $RECEIVE file.
- Section 21, Traps and Signals, describes trap codes, TNS/R native signals, and the error lists used to report trap and signal conditions.

- Section 22, OSS Error Information, describes how to find information on Open System Services (OSS) errors.

Related Reading

This manual assumes that you are familiar with the HP system architecture and the NonStop operating system. The following manuals provide information about the hardware architecture and the operating system:

- *Introduction to Tandem NonStop Systems*
- The introductions and system description manuals for individual HP NonStop systems

While using this manual, you might need to refer to the following related programming manuals.

- *Guardian Programmer's Guide*
- *Guardian Procedure Calls Reference Manual*
- *Guardian Programming Reference Summary*
- *Open System Services System Calls Reference Manual*
- *Open System Services System Calls Reference Manual*

Notation Conventions

General Syntax Notation

The following list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

```
MAXATTACH
```

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

`file-name`

computer type. `Computer type` letters within text indicate C and Open System Services (OSS) keywords and reserved words. Type these items exactly as shown. Items not enclosed in brackets are required. For example:

```
myfile.c
```

italic computer type. `Italic computer type` letters within text indicate C and Open System Services (OSS) variable items that you supply. Items not enclosed in brackets are required. For example:

```
pathname
```

[ ] Brackets. Brackets enclose optional syntax items. For example:

```
TERM [\system-name.]$terminal-name
```

```
INT[ERRUPTS]
```

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list may be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
LIGHTS [ ON             ]
       [ OFF            ]
       [ SMOOTH [ num ] ]
```

```
K [ X | D ] address-1
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list may be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name }
                  { $process-name  }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

… Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address-1 [ , new-value ]...
```

```
[ - ] {0|1|2|3|4|5|6|7|8|9}...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be entered as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;
```

```
LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate that the symbol is a required character that you must enter as shown. For example:

"[" *repetition-constant-list* "]"

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

CALL STEPMOM (*process-id*) ;

If there is no space between two items, spaces are not permitted. In the following example, there are no spaces permitted between the period and any other items:

$*process-name*.#*su-name*

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

ALTER [/ OUT *file-spec* /] CONTROLLER

   [, *attribute-spec*]...

Notation for Messages

The following list summarizes the notation conventions for the presentation of displayed messages in this manual.

Nonitalic text. Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

```
Backup Up.
```

lowercase italic letters. Lowercase italic letters indicate variable items whose values are displayed or returned. For example:

p-register

process-name

[ ] Brackets. Brackets enclose items that are sometimes, but not always, displayed. For example:

Event number = *number* [Subject = *first-subject-value*]

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list might be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

LDEV *ldev* [CU %*ccu* | CU %...] UP [(*cpu*,*chan*,%*ctlr*,%*unit*)]

{ } Braces. A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list might be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LBU { X | Y } POWER FAIL
```

```
process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown.          }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign. A percent sign precedes a number that is not in decimal notation. The %þnotation precedes an octal number. The %Bþnotation precedes a binary number. The %Hþnotation precedes a hexadecimal number. For example:

```
%005400
```

```
P=%p-register E=%e-register
```

Notation for Management Programming Interfaces

The following list summarizes the notation conventions used in the boxed descriptions of error lists in this manual.

UPPERCASE LETTERS. Uppercase letters indicate names from definition files; enter these names exactly as shown. For example:

```
ZCOM-TKN-SUBJ-SERV
```

lowercase letters. Words in lowercase letters are words that are part of the notation, including Data Definition Language (DDL) keywords. For example:

```
token-type
```

Change Bar Notation

Change bars are used to indicate substantive differences between this edition of the manual and the preceding edition. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

> The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).
>
> The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to docsfeedback@hp.com.

Include the document title, part number, and any comment, error found, or suggestion for improvement you have concerning this document.

1 Introduction

This manual describes the Guardian messages associated with the NonStop operating system. This manual is dual-threaded, covering both G-series and H-series releases. For each message, the description provides an explanation of the cause, a discussion of the effect on the system, and suggestions for corrective action or response.

Several types of messages can be returned to your program:

- Procedure errors
- Interprocess messages
- Error lists
- Traps and signals

This introductory section explains each of these message categories.

Note that Sections 2-18 describe the messages associated with specific procedures. Each of these sections describes procedure errors or error lists or both. Section 19 and Section 20 describe interprocess messages. Section 21 describes traps and signals. Section 22 describes how to find more information on Open System Services (OSS) errors.

Procedure Errors

Procedure errors are values returned to your program after your program calls procedures such as file-system procedures. Your application program should test for possible errors and take appropriate action when necessary.

Your program detects procedure errors in two ways, depending on the procedure being called. Errors returned by D-series and later procedures are not in the same format as those returned by C-series procedures. The names of D-series and later procedures can be recognized by their trailing underscore; the names of most C-series procedures do not contain the underscore character.

G-Series and Later Procedure Errors

Many G-series and later procedures return procedure error values through an output parameter of the procedure. For these procedures, you simply check the value of the error or status parameter to determine whether an error occurred. For example, an error returned in the *error* parameter of the procedure FILE_CREATE_ is a simple file-system error code giving the status of the operation.

Some G-series and later procedures return additional information through an *error-item* or *error-detail* parameter. Depending on the procedure, this information can supply the ordinal number of the parameter in error or other detailed information about the error encountered.

All G-series and later operating-system procedures have an integer returned value called *error* or *status*. If the error might have associated information, the extra information is returned in an integer *error-detail* output parameter. If *error* is a returned value, it is always possible to obtain it or return it. Bounds checking is not required.

When *error-detail* is used, most *error* values share the error designations shown in Table 1-1.

Table 1-1. Error Designations

Value	Meaning
0	Successful operation
1	File-system error other than 0, 2, or 3; *error-detail* contains the file-system error number
2	Parameter error
3	Bounds error on reference parameter
4 - *n*	Other error; *error-detail* can contain additional information

Four G-series file information procedures are available for checking on error conditions:

- FILE_GETINFO_
- FILE_GETINFOBYNAME_
- FILE_GETINFOLIST_
- FILE_GETINFOLISTBYNAME_

These procedures are described in the *Guardian Procedure Calls Reference Manual*. See Section 2, File-System Errors, for information regarding their use in obtaining file-system error information.

C-Series Procedure Errors

C-series procedures continue to use condition codes; condition codes are not used by D-series and later procedures. A condition code of“equal”(CCE) indicates success; a condition code of “less than” (CCL) or “greater than” (CCG) indicates that a problem occurred and that the error should be retrieved. The completion of a read associated with a C-series system message returns a condition code of “greater than” (CCG) and file-system error 6 from the FILEINFO procedure.

As has been the case for previous releases of the operating system, you use FILEINFO to obtain the error code. You should use the file number form of the FILEINFO procedure to obtain information about errors relating to operations involving open files; the last error associated with the file number is returned.

A typical method of testing for an error and calling FILEINFO to get the error value might appear as follows:

```
IF <> THEN      ! If CCL or CCG occurred, then
CALL FILEINFO   ! call this procedure to get the error value.
```

Interprocess Messages

Interprocess messages are data structures that are exchanged between processes or between the operating system and a process. Interprocess message are received by a user process through its $RECEIVE file. A user process sends an interprocess message by opening the process to which the message is to be sent.

This manual describes two types of interprocess messages:

- Interprocess command interpreter messages
- System messages

Interprocess Command Interpreter Messages

An interprocess command interpreter message is an interprocess message that is exchanged between the command interpreter and an application process. The Tandem Advanced Command Language (TACL) command interpreter is supplied by HP for use on the operating system.

Use of interprocess command interpreter messages is documented in the *Guardian Programmer’s Guide*.

System Messages

A system message is an interprocess message that is sent from the operating system to an application process. Use of these messages is documented in the *Guardian Programmer’s Guide*.

Error Lists

Error lists are Subsystem Programmatic Interface (SPI) buffers returned to an application program by another process. Error lists can be returned to your application process if you are using SPI to send requests to another process. If the other process encounters a procedure error, it returns the error information in an error list to your application process.

Error List Content

Unlike the error messages described elsewhere in this manual, error list information is returned in the form of tokens that are meaningful to application processes rather than in the form of displayable text. The content of error lists can vary; for example:

- A response record can contain both response data and error or warning information.
- A single response record can contain multiple error lists, especially if warnings occur.

- A single error can actually be a pass-through error, an error that originated in another subsystem that was called by the subsystem to which the command was sent.

Figure 1-1 shows the format of an error list.

Figure 1-1. Error List Format

header	ZSPI-TKN-RETCODE	ZSPI-TKN-ERRLIST	*n*	*error token*	*error token*	ZSPI-TKN-ENDLIST

The value of the return token, ZSPI-TKN-RETCODE, indicates the content of the error list:

- If zero and no error list follows, no error occurred.
- If zero but one or more error lists follow, the error lists contain only warning messages: unusual conditions that might be of interest to the requester but do not prevent the server from completing the command.
- If nonzero, the error list must contain at least one token with an error number that matches the value of the return token. Each subsystem defines its own set of error numbers.

Error Lists and Non-SPI Subsystems

Some HP software facilities do not have a programmatic command interface based on SPI but do define standard error lists. If a HP subsystem that has an SPI-based command interface calls one of these facilities and receives an error, the subsystem usually tries to recover from the error. If that is not possible, the subsystem reports the error in an SPI error token (ZSPI-TKN-ERROR), embeds it in an error list of the prescribed form, and encloses this error list in its response.

For information about creating error lists, additional information about tokens and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

Traps and Signals

Certain critical problems can cause a process to be unable to continue executing. These are typically the result of coding errors, but other conditions, such as the lack of a system resource (for example, memory), can also prevent normal process execution. Such conditions are reported as traps to TNS processes and as signals to TNS/R native processes. A trap is a software mechanism that stops process execution. A signal is a software interrupt that can notify a process of other events, such as timer expiration, as well as of critical error conditions.

The set of signals that are used in the Guardian environment is known as TNS/R native signals. This set is a subset of a larger set of signals used in the Open System Services (OSS) environment. Most of the TNS/R native signals are caused by the same conditions that cause traps to occur in TNS processes. In this manual, equivalent trap and signal conditions are described together.

Each description of a trap or signal condition contains the following information:

- The signal name or trap number
- The cause of the trap or signal
- The effect of the error on the system
- The recovery actions you can take

You can use the ARMTRAP procedure to specify a location in your application program where execution should begin if a trap occurs. Your program can use information passed to investigate the cause of the error. You can even reset the trap mechanism and cause the program to restart.

Use of the ARMTRAP procedure and user-written trap processes is documented in the *Guardian Programmer's Guide*.

2 File-System Errors

Error Codes

The file system of the NonStop operating system returns a code to the calling procedure to indicate errors and other special conditions. Because most programs use the file system, these conditions can occur during execution of almost any user-written application or any program supplied by HP.

Many programs display file-system error codes on the user's terminal. These messages typically contain the code number and a short message. For example:

```
WARNING - $VOLUME.SUBVOL.FILE   ERR 11
```

To find out what the file-system error means, look up the explanation in this section under the error code number.

You can obtain a short explanation of any file-system error code or OSS errno value at the TACL prompt by entering:

```
ERROR number
```

If you want to scan a list of all the errors, enter:

```
ERROR -1
```

Table 2-1 lists the major categories of error numbers.

Table 2-1. File-System Error Categories

Number	Category
0	No error: The operation executed successfully. For C-series procedures, the condition code is set to "equal to" (CCE).
1-9	Warning: The operation has executed with the exception of the indicated condition. For warning 6, data is returned in the application process buffer. For C-series procedures, the condition code is set to "greater than" (CCG).
10-255, 512-32767	Error: The operation encountered an error or a special condition that the application must recognize, for instance, an aborted transaction on an audited TMF file. For C-series procedures, the condition code is set to "less than" (CCL).
300-511 (except 538)	Application-defined error: These error numbers are reserved for use by application processes.
538	FILE^CHECK procedure: This procedure could not return the primary or secondary extent value because the extent size is greater than 65535 pages.
4000-4999	Open System Services error. For information on how to find the meaning of an OSS error, see Section 22, OSS Error Information.
5000-5999	NonStop Storage Management Foundation (SMF) error.

C-Series and D-Series Error Handling

When an operating-system procedure call returns an error, it is sometimes necessary to call a second procedure to obtain additional information about the error. Some C-series procedures return only condition codes and require that a second procedure be called to obtain the file-system error code; other procedures might require different information to handle the error properly.

When using C-series procedures, additional information about an error on a file is obtained by calling the FILEINFO procedure. The information returned includes the type of device associated with the error if you include the *devtype* parameter in the FILEINFO call and if a device type is applicable. (For some errors returned by subsystems such as the TMF subsystem, there is no associated physical device.) Refer to Appendix A in the *Guardian Procedure Calls Reference Manual* for a list of the device types and subtypes.

On D-series and later releases, file-system error information is also available using one of the procedures described in Table 2-2.

Table 2-2. D-Series Error Information Procedures

Procedure	When to Use
FILE_GETINFO_	To obtain information about a file using the file number (*filenum*).
FILE_GETINFOLIST_	To obtain information about a file using the file number (*filenum*).
FILE_GETINFOBYNAME_	To obtain information about a file using the file name (*file-name*). If you call FILE_GETINFOBYNAME_ in a waited manner, the system returns the information in the procedure output parameters. However, if you call this procedure in a nowait manner, the system returns the information in the system message -108 (nowait FILE_GETINFOBYNAME_ completion), which is described in Section 20, System Messages.
FILE_GETINFOLISTBYNAME_	To obtain information about a file using the file name (*file-name*).

For more information on errors related to terminals, line printers, tape drives, card readers, interprocess communication, and the operator console, refer to the *Guardian Programmer's Guide*. For more information about interprocess communication errors that can occur when applications use the Subsystem Programmatic Interface (SPI), refer to the *SPI Programming Manual*.

File-system errors returned by the data communications subsystems can have special meanings depending on the particular subsystem, access method, or protocol being used. After obtaining the device type associated with the error, consult Appendix A of the *Guardian Procedure Calls Reference Manual* to determine which subsystem is involved and refer to the appropriate manual for the subsystem.

Note. In the error code descriptions, if a device-type number includes a dot (.), the digits to the left of the dot are the device type and the digits to the right of the dot are the device subtype.

Error Mapping to C Language errno Values

Some Guardian file-system errors map to corresponding C language error numbers. These error numbers are returned in the `errno` value when using the C programming language. See Table 2-3 for a list of file-system errors and their corresponding C `errno` values.

Table 2-3. Guardian File-System Errors Mapped to C Errors (page 1 of 2)

File-System Error	C errno Symbolic Name
0-1	ENOERR
2	ENOENT
4	EIO
11	ENOENT
12	EGUARDIANOPEN
13-15	ENOENT
16	EBADF
18	ENOENT
21	EINVAL
22	EFAULT
24	EPERM
29	EINVAL
31-37	ENOMEM
43-44	ENOSPC
45	EFBIG
46	EIO
48	EACCES
49	EINVAL
50-51	EIO
53	EFSERR
54	EINVAL

Table 2-3. Guardian File-System Errors Mapped to C Errors (page 2 of 2)

File-System Error	C errno Symbolic Name
55	EIO
59	EFILEBAD
60	EWRONGID
61	EIO
73	EGUARDIANLOCKED
103	EIO
232	EIO
549, 560, 564, 590	EINVAL

Error Code Descriptions

```
0       (%0)     The operation completed successfully.
```

Cause. The file-system procedure successfully completed the requested operation.

Effect. The file-system procedure returns to the calling procedure.

Recovery. Informative message only; no corrective action is needed.

```
1       (%1)     A read procedure reached end-of-file or a
                 write procedure reached end of tape.
                 (device type:  3, 4, or 6)
```

Cause. A read procedure encountered a logical or physical end-of-file indicator, or a write procedure reached the end of the physical medium.

Effect. The procedure returns.

Recovery. Recovery, if any, is application dependent.

```
2       (%2)     The operation specified is not allowed on
                 this type of file.  (device type:  any)
```

Cause. The procedure is invalid for the given type of file. For example:

- PURGE of a nondisk file
- KEYPOSITION on an unstructured disk file
- Invalid SETMODE operation
- ACTIVATERECEIVETRANSID, READUPDATE, or REPLY attempted on $RECEIVE opened with a *`receive-depth`* of zero.

This error is also returned from REPLYX in any of the three following cases:

- The address of a parameter is extended, but no segment is in use at the time of the call or the segment in use is invalid.
- The address of a parameter is extended, but is an absolute address.
- The file system cannot use the user’s segment when needed.

Effect. The operation might return without performing all or part of the requested operation. For example, FILE_GETINFOLIST[BYNAME]_ can return without all of the requested information.

Recovery. Correct the request.

```
3       (%3)     An open or purge of a partition failed.
                 (device type:  3)
```

Cause. An OPEN or FILE_OPEN_ operation could not open one or more partitions defined for the file, or a purge or other operation for a partitioned file could not operate on one or more of the defined partitions.

Effect. The open operation succeeds. However, any future attempts to access the nonexistent partition return an error 72 (attempt to access unmounted partition). The purge operation deletes what it can.

Recovery. Correct the operation or parameters.

```
4       (%4)     An open operation for an alternate key file
                 failed.  (device type:  3)
```

Cause. An open operation for a structured file with alternate keys could not open an alternate-key file.

Effect. The open succeeds. However, any subsequent attempts to access the file by way of an alternate key stored in the unopened file result in file-system error 46 (invalid key specified). When an insertion or update is made to a file having an unopened alternate-key file, the insertion (that is, the write) finishes successfully but does not update the unopened alternate-key file.

Recovery. Corrective action is application dependent.

```
5       (%5)     Sequential buffering not used because no
                 space was available in the PFS for the
                 sequential block buffer.  (device type:  3)
```

Cause. Space was not available in the process file segment (PFS) for the sequential block buffer or the file is not a structured disk file.

Effect. Sequential-block buffering is not used.

Recovery. Informative warning message only; no corrective action is needed.

```
6       (%6)     A system message was received from another
                 process.  (device type:  2)
```

Cause. The process received a system message from the operating system.

Effect. Data is returned in the application process buffer.

Recovery. Informative message only. The process receiving the system message should read it (from $RECEIVE) to determine what action is needed.

```
7       (%7)     This process cannot accept a CONTROL,
                 CONTROLBUF, SETMODE, or RESETSYNC because
                 $RECEIVE is not opened correctly.
                 (device type:  0)
```

Cause. The process receiving the error message called CONTROL, SETMODE, RESETSYNC, or CONTROLBUF for a server process file, but the server process did not open its $RECEIVE file to enable receipt of these messages either with the procedure OPEN parameter *`flags`*.<1> set to 1 or with the FILE_OPEN_ procedure parameter *`options`*.<15> set to 0.

Effect. The procedure returns without performing the requested operation.

Recovery. Either have the server process open $RECEIVE with the OPEN parameter *`flags`*.<1> set to 1 or FILE_OPEN_ parameter *`options`*.<15> set to 0, correct the file parameter on the procedure call, or eliminate the call.

```
8       (%10)    The operation was successful (examine MCW
                 for additional status).
                 (device type:  11.40, 11.42)
```

Cause. An operation to an EnvoyACP/XF data-communication line (synchronous data link control or advanced data communications control procedures) finished successfully.

Effect. The procedure returns additional status information in the message control word (MCW).

Recovery. Retrieve information from the MCW before proceeding.

```
9       (%11)    A read through a locked record was
                 successful.  (device type:  3)
```

Cause. A “read-thru-locks with warning” mode was specified, and the record returned was locked by another user.

Effect. The procedure returns the requested data and the error code.

Recovery. Informative message only; no corrective action is needed.

```
10      (%12)    The new record or file could not be created
                 because a file by that name or a record with
                 that key already exists.  (device type:  3)
```

Cause. A process requested creation of a new disk file, insertion of a new record in a file, or insertion of a new record with a unique alternate key in a structured file, but a file by that name or a record with that key already exists.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
11      (%13)    The file is not in the directory or the
                 record is not in the file, or the specified
                 tape file is not on a labeled tape.
```

Cause. An operation referred to a nonexistent disk file or record, or the indicated tape file was not on a labeled tape.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent; for “record not in file,” it depends on the positioning mode.

```
12      (%14)    The file is in use.  (device type:  any)
```

Cause. The specified file was in use, with exclusive or protected access, by another process.

If a backup open is being performed and the file number is currently opened by the backup process, the open operation returns this file-system error number though the file was not in use.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
13      (%15)    The filename was not specified in proper
                 form.  (device type:  any)
```

Cause. The specified file name was not in proper form.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the file name.

```
14      (%16)    That device does not exist on this system.
                 (device type:  any)
```

Cause. The specified device or process did not exist on the system.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the device name or process name.

```
15      (%17)    The volume or system specified for a file
                 RENAME operation does not match the name of
                 the volume or system of the file.
                 (device type:  3)
```

Cause. The disk volume name or system number specified for a FILE_RENAME_ or RENAME procedure did not match the name of the volume or system number on which the given file resides.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the device name or process name.

```
16      (%20)    No file with that file number has been
                 opened.  (device type:  any)
```

Cause. A file number was supplied to a file-system procedure call but no file with that number was open.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Open the file using either the FILE_OPEN_ procedure or the OPEN procedure; then make your other calls such as FILE_GETINFO_ or FILEINFO.

```
17      (%21)    A paired-open was specified and the file is
                 not open by the primary process, the
                 parameters supplied do not match the
                 parameters supplied when the file was opened
                 by the primary, or the primary process is
                 not alive.  (device type:  any)
```

Cause. A backup process attempted an OPEN or FILE_OPEN_ operation, or the primary process attempted a CHECKOPEN or FILE_OPEN_CHKPT_ operation, but either the file was not opened by the primary process, the parameters supplied did not match the parameters supplied when the file was opened by the primary, or the primary process was not alive.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
18      (%22)    The system specified does not exist in the
                 network.  (device type:  any)
```

Cause. The specified system does not exist in the network; no connection has been made to this system since the last cold load.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the system name, or make sure that the lines between the local system and the referenced system are up.

```
19      (%23)    There is no more space for devices in the
                 logical device table.  (device type:  any
                 except 1 and 2)
```

Cause. The logical device table was full, so the file system could not add the specified device or process.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
20      (%24)    File name is too long to be opened across a
                 network, or swap file or system name is not
                 acceptable.  (device type:  any except 2)
```

Cause. An unconverted process tried to gain network access either to a process that has a name of more than five characters or to a device that has a name of more than seven characters. Error 20 also occurs if the opener has a process name of more than five characters or a home terminal name of more than seven characters when using a procedure that requires names in internal format. Error 20 can also indicate an incorrect swap file or system designation.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. If the opener's process name is too long, run it under a shorter name. If the opener's home terminal name is too long, either run the process with the TACL RUN command specifying "TERM *name*"; run the application from a terminal with a shorter name, rename the terminal in the SYSGEN configuration file, and regenerate the system. This problem should not occur if you use the procedures introduced in the D-series release (such as FILE_OPEN_), because they do not use names in internal format.

```
21      (%25)    An illegal <count> was specified in a
                 file-system call, or the operation attempted
                 to transfer too much or too little data.
                 (device type:  any except 2)
```

Cause. A process specified an invalid `count` parameter in a file-system call, or the operation tried to transfer too much or too little data. Typical causes are:

- A FILE_CREATE_ or CREATE procedure call for a structured disk file contains an invalid record length or alternate-key length.
- A FILE_CREATE_ or CREATE procedure call for a disk file requested a size larger than the system limit.
- A file access specifies an inconsistent key length or compare length.
- A disk I/O request spans more than two extents, is greater than 4096 bytes, or exceeds eight sectors, or a request under long transfer mode is not a multiple of 2048.
- For a 5520 printer request, the size of the direct-access, vertical-format file buffer is not an even number or is greater than 508 bytes, a line length exceeds the maximum transfer size for the current print mode, or a print line is too long for the specified printer width.
- A data line written to the Spooler is too long.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. This is a coding error; corrective action is device dependent and application dependent.

```
22      (%26)    The application parameter or buffer address
                 is out of bounds.  (device type:  any)
```

Cause. A process specified an out-of-bounds parameter or a buffer address parameter in a file-system call; that is, a pointer to the parameter or the buffer has an address that is greater than the memory associated with the data area of the process or that is in the part of the stack used by the file system.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. This is a coding error; corrective action is application dependent.

```
23      (%27)    The disk address is out of bounds, or the
                 maximum number of blocks in an alternate-key
                 file is exceeded.  (device type:  3)
```

Cause. A disk address specified in a file-system call was too large or too small, or the maximum number of blocks in an alternate-key file was exceeded. This error indicates corrupt data or a corrupt alternate-key file.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent. However, you might need to increase the workfile size.

```
24      (%30)    Privileged mode is required for this
                 operation.  (device type:  any)
```

Cause. A nonprivileged user or process attempted to perform an operation requiring privileged mode. The OPEN procedure returns this error when *flags*.<1> is set to 1 and the file being opened is not $RECEIVE. The FILE_OPEN_ procedure returns this error when either *options*.<14> or *options*.<15> is set to 1 and the file being opened is not $RECEIVE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. If you intended to use the privileged feature, have the system manager license the program file.

If a call to OPEN returned this error, check the *flags* parameter. If a call to FILE_OPEN_ returned this error, check the *options* parameter.

```
25      (%31)    AWAITIO[X] or CANCEL attempted on file
                 opened for waited I/O.  (device type:  any)
```

Cause. A process called AWAITIO[X] or CANCEL for a file opened for waited I/O.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Either open the file for nowait I/O, correct the file number, or make another correction appropriate to the application.

```
26      (%32)    AWAITIO[X] or CANCEL or CONTROL 22 attempted
                 on a file with no outstanding I/O requests.
                 (device type:  any)
```

Cause. A process called AWAITIO[X], CANCEL, or CONTROL 22, but no I/O requests were outstanding on the file; or a process called FILE_COMPLETE_ with a *timeout* value of -1 (wait indefinitely) but no I/O had been initiated.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
27      (%33)    An operation was attempted with outstanding
                 no-waited I/O requests pending.
                 (device type:  any)
```

Cause. A process tried a waited operation on a file that was opened for nowait I/O, and outstanding nowait I/O requests were pending on that file.

(Note that some operations cannot be performed nowait, such as SETMODE, POSITION, KEYPOSITION, or SETPARAM.)

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
28      (%34)    The number of outstanding nowait operations
                 would exceed that specified; an attempt was
                 made to open a disk file or $RECEIVE with
                 the maximum number of concurrent operations
                 more than 1; an attempt to ADD more than the
                 configured maximum number of subdevices for
                 an I/O process; or sync depth exceeds number
                 the opener can handle; or trying to run more
                 than 254 processes from the same object
                 file.  (device type:  any)
```

Cause. This error can occur for any of the following reasons:

- A nowait I/O request brought the number of outstanding nowait requests on the given file to a value greater than the maximum nowait depth specified when the file was opened.
- An attempt was made to open a disk file or the $RECEIVE file with a maximum number of concurrent nowait operations greater than 1.
- An attempt was made to add more than the configured maximum number of subdevices for an I/O process.
- The sync depth exceeds the number the opener is prepared to handle.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
29      (%35)    A required parameter is missing in a
                 procedure call, or two mutually exclusive
                 parameters were supplied.
                 (device type:  any)
```

Cause. A file-system procedure call was missing a required parameter or supplied two mutually exclusive parameters. The *`string:maxlen`* and *`actual-length`* parameters must both be present if either one is present. If one of these two parameters is present but the other is missing, this error is returned.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. This is a coding error; correct the call.

```
30      (%36)    Message system is unable to obtain memory.
                 (device type:  any except 2)
```

Cause. This message indicates that the system has run out of one of the following resources:

- No entry was available in the message block pool to perform the specified operation.
- A process is already using its maximum number of RECEIVE message blocks.
- A process is already using its maximum number of SEND message blocks.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Check the system for processes that use too many messages, or for processes whose message limits are too low. (A process can change its limits by using the CONTROLMESSAGESYSTEM procedure call.) Retry the operation. Try to determine what system process or application process is causing the fault. If the problem persists, contact your HP representative.

```
31      (%37)    Unable to obtain file-system buffer space.
                 (device type: any)
```

Cause. This message is returned for a privileged operating system call. Insufficient space was available in the process file segment (PFS) for a file-system buffer needed to perform the specified operation.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Check the program to see if it uses too much buffer space, opens too many files, or uses too many DEFINEs. Try reducing the workload assigned to the process.

```
32      (%40)    Unable to obtain storage pool space (SYSPOOL
                 or EXTPOOL).  (device type: any)
```

Cause. This message is returned for a privileged operating system call. Insufficient space was available for a control block needed to perform the specified operation (perhaps due to exceptionally long control blocks).

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. For an insufficient space error, wait, then try again; check the system for processes that are using too much memory.

If the problem persists, contact your HP representative. Configuration information for the current system image should be saved for your HP representative.

```
33      (%41)    I/O process is unable to obtain sufficient
                 buffer space.  (device type:  any except 2)
```

Cause. Insufficient buffer space was available for the I/O process. This message is returned for a privileged operating system call.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again; check the system for processes that are using too much memory for I/O or applications attempting to queue very large I/O operations. For persistent network errors, provide more buffer space for the network process. If the disk process ran out of lock space, try to reduce the number of concurrent locks on this disk process.

If the problem persists, contact your HP representative.

```
34      (%42)    Unable to obtain a file-system control
                 block.  (device type:  any)
```

Cause. All file-system control blocks are in use. This message is returned for a privileged operating system call.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again. Check the system for processes that use too many open files.

If the problem persists, the system has run out of some resource. Contact your service provider.

```
35      (%43)    Unable to obtain an I/O process control
                 block, or the transaction or open lock unit
                 limit has been reached.
                 (device type:  any except 2)
```

Note. The above error description is displayed only on systems running J06.08 and earlier J-series RVUs and H06.19 and earlier H-series RVUs.

```
35      (%43)    Unable to obtain a lock control block, or
                 the transaction or open lock unit limit has
                 been reached. (device type: any except 2)
```

Note. The above error description is displayed only on systems running J06.09 and later J-series RVUs and H06.20 and later H-series RVUs.

Cause. All I/O process control blocks are in use, or a requester tried to acquire too many record locks or file locks. This message is returned for a privileged operating-system call.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again. Check the system for processes that are performing too many concurrent I/O operations, or rewrite the application to request fewer locks. If the problem persists, contact your service provider.

```
36      (%44)    Unable to lock physical memory; not enough
                 memory available.  (device type:  any)
```

Cause. Insufficient physical memory was available to perform the specified operation.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again. If the problem persists, check the system for processes that use too much locked memory; run MEASURE to help determine which processes are locking memory.

```
37      (%45)    I/O process is unable to lock physical
                 memory.  (device type:  any except 2)
```

Cause. Insufficient physical memory was available to perform the specified I/O operation. This message is returned for a privileged operating system call.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again. Check the system for processes that use too much locked memory; run MEASURE to help determine which processes are locking memory. If the problem persists, contact your service provider.

```
38      (%46)    Attempt to perform operation on wrong type
                 of TNS system.  (device type:  any except 2)
```

Cause. Program running on a NonStop system specified an operation available only on a NonStop 1+ system.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Recode the application to eliminate the invalid operation.

```
39      (%47)    The server process received a request with a
                 sync ID older than the set of saved replies.
                 (device type:  0)
```

Cause. A sync ID was encountered that is older than the current sync ID minus the sync depth.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Recovery is application dependent.

```
40      (%50)    The operation timed out.  AWAITIO[X] or
                 FILE_COMPLETE_ did not complete within the
                 time specified by its <time-limit>
                 parameter.
```

Cause. One of the following occurred:

- A call to AWAITIO[X] or FILE_COMPLETE_ did not finish within the time specified by its `timelimit` parameter.
- A message was added to the caller's queue (such as a BREAK message) while the procedure PROCESS_DEBUG_ or DEBUGPROCESS was being used.

Effect. For AWAITIO[X], if 0D was specified for the time limit (completion check) or if -1 was specified for the file number (wait for completion of an operation on any file), the operation did not finish. If a nonzero time limit and a particular file were specified, the operation timed out and was automatically canceled; it is considered completed.

For FILE_COMPLETE_, this error is only returned when a time limit other than -1D (wait indefinitely) was specified and the operation did not finish.

The requested action might have been performed.

Recovery. Corrective action is application dependent and depends on whether a read or a write was involved or a PROCESS_DEBUG_ or DEBUGPROCESS operation was being used.

```
41      (%51)    A checksum error occurred on a file
                 synchronization block.  (device type:  3)
```

Cause. A file synchronization block was in error, probably because a user program modified the file-system sync buffer area.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the coding error.

```
42      (%52)    Attempt to read from unallocated extent.
                 (device type:  3)
```

Cause. A process tried a read operation when the file was positioned to an unallocated disk address.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the coding error.

```
43      (%53)    Unable to obtain disk space for file extent.
                 (device type:  3)
```

Cause. The specified volume did not contain enough contiguous free space to permit allocating an extent of the size needed for the file or for the directory.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Purge unneeded files residing on the volume.

Determine the amount of available space and the number and size of fragments on the volume by using the DSAP FREESPACE command or (on D-series releases only) the LISTFREE command. If there is excessive fragmentation on the disk, have the operator consolidate the available space by using the Disk Compression Program (DCOM).

For a temporary workaround, specify a different *swapvol* in the RUN command.

```
44      (%54)    The disk directory or DCT is full.
                 (device type:  3)
```

Cause. The disk directory is full. For a named process, the destination control table (DCT) is full.

Effect. The procedure sets the error code and returns without adding the entry.

Recovery. For disk files, purge some files, then try again, or have the operator relabel the disk to allow for more directory entries. The directory structure might allow you to add new files whose names correspond to other areas of the directory structure where there is still space.

For process files, the system might not create any newly named process until at least one existing named process has stopped.

```
45      (%55)    The file is full; or two entries for
                 <process-name> were already in the PPD.
                 (device type:  3)
```

Cause. This error can occur for the following reasons:

- The file system could not add any more records to the given disk file.
- During a call to PROCESS_CREATE_ , NEWPROCESSNOWAIT, or NEWPROCESS, two entries (both the primary and the backup) for *process-name* were already in the process-pair directory, so no new entry could be added.

Effect. For unstructured disk files, as much data as possible is written.

Otherwise, the procedure sets the error code and returns without adding to the file.

Recovery. Either create a new file with larger extents and reload it, or increase the current file's maximum extents by issuing, for example, the FUP ALTER command with the MAXEXTENTS option.

For a PROCESS_CREATE_ , NEWPROCESS, or NEWPROCESSNOWAIT call, use another process name.

```
46      (%56)    An invalid key was specified; key length
                 passed to CREATE exceeds 255 bytes; or
                 application failed to open an alternate-key
                 file.  (device type:  3)
```

Cause. An operation on a structured file specified an invalid key, or the key length passed to CREATE exceeds 255 bytes. The application might have failed to open an alternate-key file (see file-system error 4).

Effect. The procedure sets the error code and returns without performing the specified operation.

Recovery. Corrective action is application dependent. See the *Enscribe Programmer's Guide* for details.

```
47      (%57)    The alternate key data is not consistent
                 with primary file data.  (device type:  3)
```

Cause. An error was encountered partway through a set of updates to primary and alternate files. The file system tried to undo what had been accomplished thus far, but it encountered another error.

Effect. The alternate key data is not consistent with the primary data; the file-system sets the error code and returns.

All further attempts to make the primary and alternate files consistent are abandoned.

Recovery. If you are operating under a Transaction Management Facility (TMF) transaction, abort the transaction.

If you are not operating under a TMF transaction, rebuild the alternate-key files with the FUP LOADALTFILE command.

```
48      (%60)    Security violation; illegal operation
                 attempted. (device type:  3)
```

Cause. The specified operation (read, write, execute, or purge) was invalid because of the way the file was secured or because of an invalid or nonexistent password in an Expand network environment.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Resecure the file or recode the application; if access is across a network, establish matching user IDs and remote passwords at both nodes.

```
49      (%61)    Access violation; attempt to use an
                 unexpired labeled tape for output; or a
                 mismatch between a DEFINE Use attribute
                 (input or output/extend) and the current
                 operation (read or write).
                 (device type:  any except 2)
```

Cause. The type of access (read, write, or execute) specified by the calling process was invalid on the given file for one of the following reasons:

- The calling process did not open the file for the type of access attempted.
- Another process had the file open in protected or exclusive mode.
- There was an attempt to use a labeled tape for output that had not expired.
- There was a mismatch between the DEFINE USE attribute and the current operation on a CLASS tape DEFINE. The DEFINE USE attribute specifies how the tape is to be used: for example, input, output, or extend.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
50      (%62)    Directory error on a disk volume.
                 (device type:  3)
```

Cause. A severe problem occurred with the directory on a disk volume. The file associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
51      (%63)    Directory on a disk volume is marked bad.
                 (device type:  3)
```

Cause. A severe problem occurred with the directory on a disk volume. The file associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
52      (%64)    Error in the disk free space table.
                 (device type:  3)
```

Cause. A severe problem occurred with the accessed disk volume. The file associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
53      (%65)    File system internal error or CP6100 file
                 management interface error.
                 (device type:  3)
```

Cause. This error can occur for the following reasons:

- There is an internal problem in the file system or I/O system.
- A process replied to an open message with a file-system error code warning in the range 1 through 9.

Effect. The procedure sets the error code and returns without continuing the requested operation.

Recovery. If this error resulted from a process reply to an open message, correct the process. Otherwise, contact your service provider. Note any console messages that result from this error. Attempt to run a CMI or SCF TRACE if the problem is repeatable.

```
54      (%66)    I/O error in disk free space table or in DP2
                 undo area.  (device type:  3)
```

Cause. A severe problem occurred with the accessed disk volume. The file associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
55      (%67)    I/O error in disk directory; the file is no
                 longer accessible.  (device type:  3)
```

Cause. A severe problem occurred on a disk volume used by the file system. The file associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
56      (%70)    I/O error on disk volume label; the volume
                 is no longer accessible.  (device type:  3)
```

Cause. A severe problem occurred on a disk volume used by the file system. The volume associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
57      (%71)    The disk free space table is full.
                 (device type:  3)
```

Cause. A severe problem occurred on a disk volume used by the file system. There might not be enough contiguous free disk space to enlarge the disk free space table.

Effect. The procedure sets the error code and returns without performing the requested operation. The file associated with the error is no longer accessible.

Recovery. Your system operator might be able to recover the file.

```
58      (%72)    The disk free space table is marked bad.
                 (device type:  3)
```

Cause. A severe problem occurred with the accessed disk volume. The file associated with the error is no longer accessible.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Your system operator might be able to recover the file.

```
59      (%73)    The disk file is bad; there is a mismatch in
                 the internal FCB, or the file structure in a
                 structured file is inconsistent.
                 (device type:  3)
```

This error can occur during an attempt to open a file, or during an attempt to read or write to a structured file. It might indicate that the file or table is broken.

If an application uses unstructured access to write to a DP2 structured file, it is possible that future attempts to read or write to this file using unstructured access will cause this error. User applications should not use unstructured access to write to a structured file.

Another example of this error occurring follows a RESTORE operation on a key-sequenced file that was backed up while it was being modified. The error eventually occurs because the file is inconsistent.

Error Occurs During File Open

Cause. The file you attempted to open is marked as corrupt. A utility command such as FUP DUP, FUP LOAD, SQLCI DUP, SQLCI LOAD, or RESTORE has not finished.

Effect. The procedure sets the error code and returns without performing the requested operation. The file or table cannot be opened.

Recovery. Wait until the utility operation finished before attempting to open the file. If the file is not currently open, then the utility abended before finishing; purge the file or table.

Error Occurs During Read/Write Activity

Cause. The structured file or table has an inconsistent structure.

Effect. The procedure sets the error code and returns without performing the requested operation. The portion of the file or table with the inconsistent structure cannot be accessed or updated.

Recovery. Examine the EMS event logs for BAD FILE DETECTED events. Run the FILCHECK utility to determine whether the file is broken. If FILCHECK reports a broken file or table, use the TMF subsystem or the RESTORE program to recover the data. If TMF or RESTORE recovery is not available, and recovery is required, contact your service provider for assistance.

```
60      (%74)    The file resides on a removed volume, the
                 device is downed or not open, or a server
                 has failed and a process has been replaced
                 by a different process with the same name
                 since the server was opened.
                 (device type:  any except 1 and 2)
```

Cause. This error can occur for the following reasons:

- The specified file resides on a volume that has been removed from the system or on a device that was brought down after the file was opened.
- The file is a device or process on a remote system whose link has gone down.
- If a server returned this error, the server received a message from a process that it does not recognize as its opener. This can happen if a FILE_OPEN_ procedure was used to open a file but the file name did not include the optional sequence number.
- Certain sequences of DDL and DML operations can result in an error 60 (lost open). Refer to the *NonStop SQL/MP Reference Manual* for more information on opening SQL files.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Ensure that the device (if any) is up, close the file and reopen it, then try again. If that is not successful, abort the operation and start over.

```
61      (%75)    No more file opens are permitted on this
                 volume or device; the system operator issued
                 a STOPOPENS command or the number of open
                 files reached the maximum allowed.
                 (device type:  3)
```

Cause. The number of open files on this volume reached the maximum allowed, or the system operator used a PUP STOPOPENS command (on D-series systems) or SCF STOPOPENS (on G-series systems) to stop any more files from being opened on the volume.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry when files can be opened.

```
62      (%76)    The volume was mounted but no mount order
                 was given, so the file open is not
                 permitted.  (device type:  3)
```

Cause. The specified disk volume was physically mounted, but the system has not received a mount request.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry the operation after issuing a mount command.

```
63      (%77)    The volume was mounted and the mount is in
                 progress, so a file open is not permitted.
                 (device type:  3)
```

Cause. The specified disk volume was physically mounted and the mount command has been given, but the mount has not finished.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry after the mount finishes.

```
64      (%100)   The volume was mounted and the mount is in
                 progress, so a file open is not permitted.
                 (device type:  3)
```

Cause. A file open operation was tried while a mount operation was in progress.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry after the mount completes.

```
65      (%101)   Only special requests permitted--or special
                 request to mirrored disk pair attempted with
                 only one device in special state.
                 (device type:  3)
```

Cause. This error can occur for the following reasons:

- A system operator designated the specified disk volume for special request mode.
- A read or write operation with the special bit set was attempted on a mirrored-disk pair that has one device in the up state, the other device in the special state.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry after the volume is available for normal operations.

```
66      (%102)   The device is downed, the LIU is not yet
                 downloaded, a hard failure occurred on the
                 controller, the disk and controller are not
                 compatible (DP1/DP2), or both halves of a
                 mirrored disk are down. (device type:  any
                 except 2)
```

Cause. This error can occur for the following reasons:

- The system operator brought down the specified device.
- A hard error occurred on the device controller.
- A path failure error in the range {210:226} occurred in a network.
- A DP2 disk process discovered a DP1 volume label or other label consistency problem.
- Both halves of a mirrored disk are down.

Effect. The procedure sets the error code and returns without performing the requested operation.

If a path failed but other lines are active, the network brings down the defective line.

If there is a DP2-DP1 label consistency problem, the disk goes down, but the system does not freeze.

If the problem occurs just after a cold load, $SYSTEM goes down, and then the monitor process cannot open the OSIMAGE file and the operator process cannot open the log file. Console messages to this effect are issued and, a minute later, the system freezes with code %002002.

Recovery. If there is a DP2-DP1 consistency problem, correct the problem.

For other devices, retry the failed operation after the device comes up. If the problem recurs, notify your system operator.

```
70      (%106)   Continue the file operation.
                 (device type:  0)
```

Cause. To indicate that the process file is ready to receive ASSIGN and PARAM messages, the receiving process replies with this error after a write of the command interpreter startup message.

Effect. The initial write of the startup message is completed.

Recovery. Continue startup sequence. Send the ASSIGN and PARAM messages if appropriate.

```
71      (%107)   A duplicate record was encountered.
                 (device type:  3)
```

Cause. The specified structured file record is a duplicate.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
72      (%110)   An attempt was made to access an unmounted
                 or nonexistent partition, or to access a
                 secondary partition.  (device type:  3)
```

Cause. The specified operation on a structured file tried to access a nonexistent partition or a partition to which access is invalid. For instance, if $B.X is the secondary partition of $A.X, then “FUP COPY $B.X” fails with this error.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
73      (%111)   The disk file or record is locked.
                 (device type:  3)
```

Cause. The operation requested access to a locked disk file or record. This error occurs only if the calling process accessed the file in alternate locking mode.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then retry.

```
74      (%112)   Number of READUPDATES without replies
                 exceeds <receive-depth>, or
                 ACTIVATERECEIVETRANSID or REPLY called with
                 an invalid <message tag>.  (device type:  2)
```

Cause. This error can occur for two reasons:

- A process called READUPDATE to read a message from $RECEIVE, but the number of outstanding messages read but not replied to equals the receive depth.
- A process called ACTIVATERECEIVETRANSID or REPLY either with an invalid message tag or when no outstanding message exists.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
75      (%113)   Requesting process has no current process
                 transaction identifier.  (device type:  3)
```

Cause. The requesting process tried to access an audited file but had no current transaction identifier, or a process called ACTIVATERECEIVETRANSID, ENDTRANSACTION, or ABORTTRANSACTION but the message specified by the *`message-tag`* has no associated transaction identifier.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
76      (%114)   Transaction is in the process of ending.
                 (device type:  3 or none)
```

Cause. The transaction was ending so it could not be aborted or resumed.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
78      (%116)   Transaction identifier is invalid or
                 obsolete.  (device type:  3 or none)
```

Cause. The transaction identifier was invalid or obsolete, or it is no longer available because the Transaction Management Facility (TMF) subsystem removed it from the system.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
79      (%117)   A transaction attempted to update or delete
                 a record which it has not previously locked.
                 (device type:  3)
```

Cause. The transaction failed to lock a record before attempting to change or delete it.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
80      (%120)   Invalid operation on audited file or
                 non-audited disk volume.  (device type:  3)
```

Cause. An invalid operation was attempted on an audited file or a nonaudited disk volume.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
81      (%121)   Operation is not valid for a transaction
                 which still has nowait I/Os outstanding on a
                 disk or process file.  (device type:  3 or
                 none)
```

Cause. The operation was invalid because the transaction had one or more outstanding nowait I/O operations on a disk or process file. Error 81 is received by ENDTRANSACTION or REPLY.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
82      (%122)   TMF not running on this system or on the
                 remote system.  (device type:  0, 3, or
                 none)
```

Cause. BEGINTRANSACTION failed for one of the following reasons:

- The Transaction Management Facility (TMF) subsystem was not running on this system.
- An I/O operation to an audited disk or a process file on a remote system was part of a TMF transaction, but the TMF subsystem was not running on the remote system.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Ensure that the TMF subsystem is running on all systems involved in the transaction, then retry the command.

```
83      (%123)   Attempt to begin more concurrent
                 transactions than can be handled.
                 (device type:  none)
```

Cause. BEGINTRANSACTION failed because the process reached its maximum number of concurrent transactions. The maximum is one transaction if the TFILE is not open; otherwise, the maximum equals the TFILE depth.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
84      (%124)   TMF has not been configured on this system
                 or on the remote system.  (device type:  not
                 applicable)
```

Cause. BEGINTRANSACTION failed for one of the following reasons:

- The Transaction Management Facility (TMF) subsystem was not configured on this system.
- An I/O operation to an audited disk or a process file on a remote system was part of a TMF transaction, but the TMF subsystem was not configured on the remote system.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then retry. Make sure the TMF subsystem is configured on all systems involved in the transaction.

```
85      (%125)   A device has not been started for TMF.
```

Cause. A device has not been started for the Transaction Management Facility (TMF) subsystem.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Ensure that devices are enabled for the TMF subsystem. For example, use the command ENABLE VOLUMES through TMFCOM.

```
86      (%126)   BEGINTRANSACTION is disabled either by the
                 operator or because one or more TMF limits
                 have been reached.
                 (device type:   not applicable)
```

Cause. BEGINTRANSACTION failed due to explicit action by an operator or because one or more audit trails have reached *maxfiles*, the maximum number of audit files permitted.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Contact the operator. If operator intervention caused the failure, have the operator correct the problem.

If one or more audit trails have reached *maxfiles*, dump audit trails until the maximum is no longer exceeded.

```
87      (%127)   Waiting on a READ request and did not get
                 it.  (device type:  10 and 60)
```

Cause. A subdevice that was waiting on a read request received an I/O request that was not a read request.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
88      (%130)   A CONTROL READ is pending so a second READ
                 is not valid.  (device type:  10 and 60)
```

Cause. A two-step read (CONTROL 22) was queued at the subdevice, so a second two-step read was invalid.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
89      (%131)   A remote device cannot accept text because
                 it has no buffer available.
                 (device type:  10)
```

Cause. The remote device cannot accept text because it does not have an available buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry the write from the application process.

```
90      (%132)   The transaction was aborted by the system
                 because its parent process died, a server
                 using the transaction failed, or a message
                 to a server using the transaction was
                 cancelled.  (device type:  3 or none)
```

Cause. The transaction aborted because either its BEGINTRANSACTION process failed, a server using the transaction failed, or a message to a server using the transaction was canceled.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Application Programmer's Guide*.

```
91      (%133)   A TMF crash occurred during commitment of
                 the transaction; the transaction may or may
                 not have been committed.  (device type:  3)
```

Cause. A serious internal error occurred on a system that runs the Transaction Management Facility (TMF) subsystem.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Contact your service provider.

```
92      (%134)   Distributed transaction aborted by system
                 because the path to a remote system that was
                 part of the transaction was down.
                 (device type:  3 or none)
```

Cause. The transaction aborted because the path to a remote system that participated in the transaction was down.

Effect. The procedure sets the error code and returns without performing the requested operation. The transaction is aborted.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
93      (%135)   A transaction was aborted because it spanned
                 too many audit trail files.
                 (device type:  3 or none)
```

Cause. The transaction was aborted because audit information filled 45% of the master audit trail since the transaction started. The Transaction Management Facility (TMF) subsystem reserves audit trail space in case the transaction must be completely backed out. If this transaction were allowed to continue past the 45% threshold, the audit trail could fill to the begin-transaction-disable level and new transactions could not start.

Effect. The procedure sets the error code and returns without performing the requested operation. The transaction is aborted.

Recovery. If this is a normal transaction that requires additional audit trail space to complete, consider temporarily increasing the audit trail capacity by increasing the number of audit trail files per volume, or by adding another active audit volume. Restart the transaction. See the *NonStop TM/MP Operations and Recovery Guide*.

```
94      (%136)   A transaction was aborted by operator
                 command.  (device type:  3 or none)
```

Cause. An operator command aborted a transaction.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. For further information, see the *NonStop TM/MP Operations and Recovery Guide*.

```
95      (%137)   A transaction was aborted because of disk
                 process takeover by backup.
                 (device type:  3)
```

Cause. The system aborted the transaction because of a processor failure that caused loss of access to a disk. This can be caused by:

- The disk volume itself down while there is a transaction outstanding against it.
- A volume cannot access its audit trail.
- The primary process of a participating disk process fails and the backup takes over.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
96      (%140)   The transaction was aborted because it
                 exceeded the AUTOABORT timeout duration.
```

Cause. The call to the ENDTRANSACTION procedure failed because the transaction was aborted by the Transaction Management Facility (TMF) autoabort function, which automatically aborts transactions that run longer than a set amount of time.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. If this is a normal transaction that requires additional time to complete, consider increasing the autoabort threshold. Do this only if you are sure that there is enough space in the audit trail to accommodate the audit trail files that may be pinned while the transaction completes. Restart the transaction. For more information, see the *NonStop TM/MP Operations and Recovery Guide*.

```
97      (%141)   Transaction aborted by call to
                 ABORTTRANSACTION.  (device type:  3 or none)
```

Cause. The transaction was aborted by a call to ABORTTRANSACTION.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
98      (%142)   Allocation of a Transaction Control Block
                 failed because the local table is full, or
                 the table on a remote system is full.
                 (device type:  0, 3, or none)
```

Cause. The transaction could not be started because:

- The local Transaction Management Facility (TMF) network active transmissions table was full.
- An I/O operation to an audited disk or a process file on a remote system was part of a TMF transaction, but the TMF network active transactions table on the remote system was full.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. See the *NonStop TM/MP Operations and Recovery Guide*.

```
99      (%143)   Process attempted to use features of a
                 microcode option that is not installed on
                 this system.  (device type:  any except 2)
```

Cause. A process tried to use features of a microcode option not installed in the system.

If this error is returned after an operation involving a TAPECATALOG DEFINE, the specific cause of the error is that the operation tried to use a CLASS TAPECATALOG DEFINE when this class of DEFINE was not supported.

Effect. The procedure sets the error code and returns without performing the requested operation.

If the cause of the error is that the operation tried to use a CLASS TAPECATALOG DEFINE, the tape request fails.

Recovery. Ensure that the system has the required microcode.

If the cause of the error is that the operation tried to use a CLASS TAPECATALOG DEFINE, you must either use a CLASS TAPE DEFINE in the operation or enable

TAPECATALOG DEFINEs. See the *DSM/Tape Catalog User’s Guide* for more information.

```
100     (%144)   Device is not ready or the controller is not
                 operational.  (device type:  any except 2)
```

Cause. This error can occur for the following reasons:

- The device was not powered up or was not online.
- A tape drive was accessed while rewinding.
- A tape drive is at a load point but not online.
- A heavily loaded processor received a call to open a server process but could not respond.
- The printer was out of paper.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Make the device ready. In the case of a heavily loaded processor, repeat the call to open the process.

```
101     (%145)   The tape is write protected.
                 (device type:  4)
```

Cause. The system could not write to the mounted tape because the tape is not write enabled.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Verify that the correct tape is in use. The method of write-enabling a tape varies depending on the specific hardware. You may need to insert a write ring or set a write protect tab on the tape volume, or you may need to change a write setting on the tape drive.

```
102     (%146)   Printer paper out, bail open or end of
                 ribbon.  (device type:  5)
```

Cause. A printer could not continue because it was out of paper or because the paper bail was not in place.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Load more paper, close the bail, or replace the ribbon as needed.

```
103     (%147)   Disk not ready due to power failure.
                 (device type:  3)
```

Cause. The disk device was not ready because the system power failed.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry the operation.

```
104     (%150)   No response from printer.
                 (device type:  5.4)
```

Cause. The printer did not return the requested status; either the printer power was off or a hardware problem occurred.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Power up the device or repair it.

```
105     (%151)   Invalid printer VFU buffer.
                 (device type:  5.4)
```

Cause. The printer direct-access vertical-format unit buffer was invalid. This error can occur for the following reasons:

- More than one stop was defined for channel 0 (top of form).
- No stops were defined for one or more channels.
- Bits <12:15> of each word were not zeros.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the programming error.

```
106     (%152)   A buffered WRITE has failed; data in printer
                 buffer was lost.  (device type:  5.4)
```

Cause. The write operation failed, and the data in the printer buffer was lost.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. The data must be generated again before it can be printed.

```
110     (%156)   Only BREAK access is permitted.
                 (device type:  6 or 61)
```

Cause. The calling process could not access the specified terminal because the user pressed BREAK, and the process had specified break mode.

Effect. The procedure sets the error code and returns without transferring any data.

Recovery. Unless the calling process uses SETMODE to indicate that its operations have break access, the terminal is inaccessible until the process processing the break calls SETMODE function 12.

If the calling process did not enable break, retry the operation, delaying at least 10 seconds between retries.

If the calling process enabled break, check $RECEIVE for the system break message and take appropriate action.

```
111     (%157)   Operation aborted because of BREAK.
                 (device type:  6 or 61)
```

Cause. The file-system procedure aborted terminal access because the user pressed BREAK before the current operation finished.

Effect. The procedure sets the error code and returns without transferring any data. Data might have been lost.

Recovery. If this process did not enable break and the error occurred during a write operation, retry the operation, delaying 10 seconds between retries.

If this process did not enable break and the error occurred during a read operation, recovery is application dependent.

Note. If more than one process is accessing a terminal when break is used, only break access is allowed after break is entered; subsequent retries are rejected with error 110 until normal access is permitted.

If this process enabled break, check $RECEIVE for the system break message and take appropriate action.

```
112     (%160)   READ or WRITEREAD preempted by operator
                 message (device type:  6) or too many user
                 console messages  (device type:  1).
```

Cause. If the device is a terminal, an operator message has preempted the requested READ or WRITEREAD operation.

If the device is the operator console, its internal buffer is full.

This error occurs only when an application process is using a terminal that is also being used as the console device.

Effect. The procedure sets the error code and returns without performing the requested operation. Any data being entered when the preemption took place is lost.

Recovery. If the device is a terminal, send a message telling the terminal operator to retype the last entry, then retry the read.

If the device is the operator console, wait, then retry.

```
113     (%161)   DEFINE class or attributes are not valid for
                 the attempted function.  (device type:  4)
```

Cause. A DEFINE was used in a manner that was not consistent with the class of the DEFINE or was not compatible with the attribute values of the DEFINE.

If this error is returned after an operation involving a TAPE DEFINE, the specific cause of the error may be that the operation tried to use a CLASS TAPE DEFINE when this class of DEFINE was not supported.

If this error is returned after an operation involving a TAPE DEFINE or TAPECATALOG DEFINE, the specific cause of the error may be one of the following:

- The DENSITY attribute was used with a specified DEVICE attribute that does not support open reel tape volumes.
- The TAPEMODE attribute was used with a specified DEVICE attribute that does not support cartridge tape volumes.
- Both the DENSITY and TAPEMODE attributes were specified.

Effect. The requested operation is ignored.

If the operation tried to use a CLASS TAPE DEFINE or CLASS TAPECATALOG DEFINE, the tape request fails.

Recovery. Examine the DEFINE class and attributes as well as the procedure parameters. See the *Guardian Programmer’s Guide* for an explanation of DEFINE classes and a discussion of how to assign values to DEFINE attributes and how to use DEFINEs in procedure calls. Change the DEFINE class or attributes or change the procedure parameters as appropriate.

If the cause of the error is that the operation tried to use a CLASS TAPE DEFINE when this DEFINE class was not supported, you must either use a CLASS TAPECATALOG DEFINE in the operation or enable TAPE DEFINEs. See the *DSM/Tape Catalog User's Guide* for more information.

If the error is due to an incompatibility among the DENSITY, TAPEMODE, and DEVICE attributes of a TAPE DEFINE or TAPECATALOG DEFINE, you must change the DEFINE in accordance with the following rules.

- Specify the DENSITY attribute only when the tape drive to be used supports open reel tape volumes.
- Specify the TAPEMODE attribute only when the tape drive to be used supports cartridge tape volumes.
- If a particular tape drive is to be used, specify the DEVICE attribute using a tape drive that matches the DENSITY or TAPEMODE attribute.
- Never specify both the DENSITY and TAPEMODE attributes in the same DEFINE.

```
114     (%162)   X25 Network problem - RESTART FAILURE.
                  (device type:  61)
```

Cause. The line will not come up.

Effect. X.25 aborts the request; an event message is sent to the EMS collector process.

Recovery. Fatal error; no recovery is possible. Determine the error from a trace of the code and try again. If the problem is still not apparent, submit the trace, log file, CONFLIST, subunit, and line configuration to your service provider.

```
115     (%163)   X25 Network problem - RESET FAILURE.
                  (device type:  61)
```

Cause. A reset attempt timed out.

Effect. X.25 aborts the request; an event message is sent to the EMS collector process.

Recovery. Fatal error; no recovery is possible. Determine the error from a trace of the code and try again. If the problem is still not apparent, submit the trace, log file, CONFLIST, subunit, and line configuration to your service provider.

```
119     (%167)   Error code value was too large to fit into
                 an 8-bit buffer;  file-system error number
                 is greater than 255.
```

Cause. The file-system error number is greater than 255 for NEWPROCESS or NEWPROCESSNOWAIT. SeeSection 5, NEWPROCESS AND NEWPROCESSNOWAIT Errors, for an explanation of the information returned in the *errinfo* parameter.

Effect. File-system error numbers greater than 255 can be used when a buffer larger than 8 bits is available.

Recovery. Determine the actual file-system error and take the appropriate action. Corrective action is application dependent. For NEWPROCESS and NEWPROCESSNOWAIT, these file-system error numbers are returned in the *errinfo* parameter.

```
120     (%170)   Data parity error, or attempt to access a
                 tape whose density is higher than the switch
                 setting on the tape drive.
                 (device type:  any except 2)
```

Cause. This error can occur for the following reasons:

- A hardware data-parity error occurred and persisted through several retries of the operation.
- The tape density is higher than the tape drive allows.

Effect. On a disk or tape READ or READUPDATE, the procedure returns as much (invalid) data as possible. You can determine the number of characters returned by checking the procedure *count-read* parameter if the procedure is a wait read or the *count-transferred* parameter of AWAITIO[X] if it is a nowait read.

Recovery. If the device is a tape drive, clean the tape head and retry the operation. If this action fails, try another tape.

Any other corrective action is device dependent and application dependent.

```
121     (%171)   Data overrun error, hardware problem.
                 (device type:  any except 2)
```

Cause. A hardware data-overrun error occurred and persisted through several retries of the operation.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. Contact your service provider.

```
122     (%172)   Request aborted due to possible data loss
                 caused by reset of circuit, CLB sequence
                 error (device type:  6, 7, 9, 11, or 61); or
                 DP2 disk process takeover (device type:  3)
```

Cause. This error can occur for the following reasons:

- A request aborted due to a possible loss of data caused by a circuit reset.
- There is a sequence error in the CIU-to-LIU bus (CLB). This is a data communications error.
- There is a DP2 process takeover.

This error occurs only on unaudited, buffered files with a sync depth equal to 0. The message is returned once for each open of the buffered file. This message also can be returned on any call that supplies a file number except for CLOSE, which never returns an error.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent and application dependent.

```
123     (%173)   Subdevice is busy.  (device type:  5, 6, or
                 10)
```

Cause. An operation was in progress on the specified subdevice.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is subdevice dependent and application dependent.

```
124     (%174)   A line reset is in progress, loss of data is
                 possible.  (device type:  6, 10, 51, 53, 59,
                 60, or 63)
```

Cause. A line reset was in progress for the subdevice or line, possibly causing a loss of data.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent and application dependent. If a data-communication process is involved, obtain a trace and save it for your service provider.

```
130     (%202)   Illegal disk address requested, or
                 formatting error occurred.
                 (device type:  3)
```

Cause. The requested address was too large for the disk space, or an error occurred during formatting.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the coding error or reformat the disk.

```
131     (%203)   Write-check error from disk; internal
                 circuitry fault.  (device type:  3.0 or 3.1)
```

Cause. The disk hardware detected an internal circuitry fault.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Notify your system operator.

```
132     (%204)   Seek incomplete from disk; cylinder address
                 not reached after retry.  (device type:  3.0
                 or 3.1)
```

Cause. The disk read-write heads did not reach the desired cylinder address after a retry.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Notify your system operator.

```
133     (%205)   Access not ready on disk; cylinder address
                 not reached.  (device type:  3.0 or 3.1)
```

Cause. The disk read-write heads did not reach the desired cylinder address.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Notify your system operator.

```
134     (%206)   Address compare error on disk.
                 (device type:  3)
```

Cause. A header search failure or header miscompare occurred on the disk. This error indicates either a request for a bad address or, possibly, a head alignment or formatting problem.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Notify your system operator.

```
135     (%207)   Write-protect violation with disk write.
                 (device type:  3)
```

Cause. An attempt was made to write to a write-protected disk.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Reset the write-protect switch to allow writes.

```
136     (%210)   Disk unit ownership error (dual-port disk).
                 (device type:  3)
```

Cause. This error can occur for the following reasons:

- The driver tried to take ownership of the drive through the controller, and the attempt failed.
- An incorrect controller address was passed to the disk process.

Effect. The disk process automatically retries.

Recovery. If the automatic retry is not successful, retry the operation yourself.

```
137     (%211)   Controller buffer parity error.
                 (device type:  any except 2)
```

Cause. A parity error occurred in the controller buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Notify your system operator.

```
138     (%212)   Interrupt overrun; a device interrupted the
                 processor before the software could respond.
                 (device type:  any except 2)
```

Cause. The device interrupted the processor before the software could respond.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again. If retries do not succeed, notify your system operator since a disk drive could go down.

```
139     (%213)   Controller error; internal diagnostic
                 failure.  (device type:  any except 2)
```

Cause. The controller failed its internal diagnostics.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Notify your system operator, or contact your service provider.

```
140     (%214)   Modem error (communication link not yet
                 established, modem failure, momentary loss
                 of carrier, or disconnect) or FOX link to an
                 EXPAND line handler is down.
                 (device type:  6, 7, 10, 11, 12, 59, 60, 61,
                 63)
```

Cause. The specific meaning of this error depends on the subsystem that returned it. Causes can include:

- The communications link is not yet established
- A modem failure occurred
- A momentary loss of carrier occurred
- The modem or link is disconnected
- The interprocessor bus monitor process ($IPB or for TorusNet configurations, $IPB`n`, where `n` is a number in the range of 1 through 4) reported that the FOX link or TorusNet vertical link to an Expand process is down
- A subunit or logical unit is not in the started condition

  Refer to the appropriate subsystem manual for additional information.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If the problem is not apparent, submit the trace, log file, CONFLIST, subunit, and line configuration to your service provider.

```
148     (%224)   Attempt to read unwritten data or to over-
                 write existing data on an optical disk.
                 (device type: 3.56)
```

Cause. There was an attempt to write to an optical disk where data was already written, or there was an attempt to read data from a location on an optical disk where there was no data.

Effect. The procedure sets the error code and returns.

Recovery. Recovery is application dependent.

```
150     (%226)   End-of-tape marker detected.
                 (device type:  4)
```

Cause. The tape unit encountered the end-of-tape marker in the forward direction during this operation, or it passed the marker on a previous operation and has not yet passed it in reverse.

Effect. The last operation is completed normally, but very little tape is left.

Recovery. Informative message only; no corrective action is needed. If this error occurs at the beginning of several tapes, contact your service provider regarding a possible hardware problem.

```
151     (%227)   Runaway tape detected, or attempt to access
                 a tape whose density is lower than the
                 switch setting on the tape drive.
                 (device type:  4)
```

Cause. The tape drive detected a runaway tape. This error usually occurs when the system tries to read a blank tape or a tape written at a density lower than the tape drive can read.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Check the tape and replace it if it is faulty. If a read fails, change the density switch setting or mount the tape on a tape drive that can read the tape at the density at which it was written. If this error occurs at the beginning of several tapes, contact your s regarding a possible hardware problem.

```
152     (%230)   Unusual end (device type:  4 or 11)--tape
                 unit went offline or CP6100 file closed
                 while requests still in progress
                 (device type:  11).
```

Cause. The device controller returned an “unusual end” status. If the device is a tape unit, the tape drive was taken offline manually during this operation.

Effect. Part of the requested operation might be complete.

Recovery. Take corrective action appropriate to the device. If this error occurs with several tapes, contact your service provider regarding a possible hardware problem.

```
153     (%231)   Tape drive power restored.
                 (device type:  4)
```

Cause. The tape-drive power failed and was restored.

Effect. The tape unit automatically returns to the ready state and to the beginning of the tape.

Recovery. If two units are set to the same unit number when the power is restored, the wrong unit might be placed online. To prevent this problem, ensure that all application programs perform a rewind and unload operation when finished with a tape unit.

```
154     (%232)   BOT detected during backspace files or
                 backspace records.  (device type:  4)
```

Cause. The tape unit encountered the beginning-of-tape (BOT) marker during a backspace-files or backspace-records operation.

Effect. Tape motion stops.

Recovery. Corrective action is application dependent.

```
155     (%233)   Only nine-track magnetic tape allowed on
                 this system.  (device type:  4)
```

Cause. A seven-track tape device was specified for an operation requiring a nine-track tape (most HP subsystems require a nine-track tape).

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry, specifying a drive configured for a nine-track tape.

```
156     (%234)   Tape command rejected, TIL or TLAM protocol
                 violation detected, or X25AM level-4 ITI
                 protocol violation.  (device type:  4, 12,
                 or 56)
```

Cause. This error can occur for the following reasons:

- The tape command was rejected.
- The procedure made a connect request (CONTROL 11) after the Tandem-to-IBM Link (TIL) was already connected or after an internal link error occurred.
- The TLAM process has entered an unrecoverable state.

For a TLAM error, a trace or processor dump is required to determine the exact nature of the problem.

Effect. The procedure aborts the call and disconnects the link. For a tape, no operation is performed.

Recovery. For a rejected tape command, check that you indicated a valid command. Inform your system manager that the tape controller is probably unloaded.

For a superfluous connect request, correct the coding error.

For an internal link error, contact your system operator.

```
157     (%235)   I/O process internal system error.
                 (device type:  any except 2)
```

Cause. An internal system error occurred.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Contact your system operator.

```
158     (%236)   Invalid function code requested for Hyper
                 Link.  (device type:  26)
```

Cause. An operation specified an invalid Hyper Link function code.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Contact your system operator or supply the correct function code.

```
159     (%237)   Device mode wrong for Hyper Link I/O
                 request, or attempt to execute PUP PRIMARY
                 command while the tape drive is waiting for
                 a labeled tape to be mounted.
                 (device type:  26)
```

Cause. This error can occur for the following reasons:

- The Hyper Link I/O process received a test request, but the I/O process was not in the test mode of operation.
- An attempt was made to issue a PUP PRIMARY command (on D-series releases) or an SCF PRIMARY command (on G-series releases), but the tape drive was waiting for a labeled tape to be mounted.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Request the test mode of operation prior to issuing test requests, or mount a labeled tape and retry the operation.

```
160     (%240)   Request is invalid for device state;
                 protocol error.  (device type:  6, 7, 10,
                 11, 51, 60, or 63)
```

Cause. A protocol error occurred. An Expand process received a PUP PRIMARY request (on D-series releases) or an SCF PRIMARY request (on G-series releases), but the backup process was not running or was misconfigured, or the Expand process could not transfer ownership to its backup for some reason.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
161     (%241)   Impossible event occurred for line state.
                 (device type:  7, 10, 11, or 60)
```

Cause. An event occurred that was impossible for the current line state; this error probably indicates a hardware problem.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
162     (%242)   Operation timed out.  (device type:  6, 7,
                 10, 11, or 60)
```

Cause. The specified operation timed out after several retries.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
163     (%243)   EOT received (device type:  7.0, 7.1, 7.2,
                 7.3, or 7.8) or power at autocall unit is
                 off (device type:  7.56 or 11).
```

Cause. Either an end-of-transmission (EOT) message was received while the procedure was waiting for a line bid or for a message, or the power at the auto-call unit was off.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
164     (%244)   Disconnect received (device type:  7.0, 7.1,
                 10, 11, 61, or 63) or data line is occupied
                 (device type:  7.56 or 11).
```

Cause. Either a disconnect was received, a send disconnect call was issued while a request was outstanding, or the data line was busy. This error message also occurs when the specified number of retries for an Expand send message are used up.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent.

```
165     (%245)   RVI received (device type:  7.0, 7.1, 7.2,
                 or 7.3) or data line not occupied after
                 setting call request (device type:  7.56 or
                 11).
```

Cause. A reverse interrupt (RVI) was received, or the data line was not occupied after setting the call request.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
166     (%246)   ENQ received (device type:  7.0, 7.1, 7.3,
                 or 7.9) or auto call unit failed to set
                 present-next-digit (device type:  7.56 or
                 11).
```

Cause. An inquiry (ENQ) was received, or the auto-call unit failed to set “present next digit.”

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
167     (%247)   EOT received on line bid (device type:  7.0,
                 7.1, 7.3, or 7.8), or data-set-status not
                 set (device type:  7.56 or 11).
```

Cause. Either an end-of-transmission (EOT) message was received in response to a line bid or selection, or “data set status” was not set after dialing all digits.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
168     (%250)   NAK received on line bid (device type:  7.0,
                 7.1, 7.3, or 7.8), or auto-call unit failed
                 to clear present-next-digit after digit-
                 present was set (device type:  7.56 or 11).
```

Cause. Either a negative acknowledgment (NAK) was received in response to a line bid or selection, or the auto-call unit failed to clear “present next digit” after “digit present” was set.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
169     (%251)   WACK received on line bid
                 (device type:  7.0, 7.1, or 7.3), auto-call
                 unit set abandon-call-and-retry
                 (device type:  7.56 or 11), or station
                 disabled or undefined (device type:  11).
```

Cause. This error can occur for the following reasons:

- A wait for acknowledgment (WACK) was received in response to a line bid or selection.
- The auto-call unit set “abandon call and retry.”
- The specified station was disabled or undefined.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
170     (%252)   No ID sequence received during circuit
                 assurance mode (device type:  7.0 or 7.1) or
                 invalid MCW entry number on write
                 (device type:  11.40).
```

Cause. This error can occur for the following reasons:

- No ID sequence was received.
- A write request for a multipoint line had an invalid message control word (MCW) entry number or an MCW entry number different from the currently active output entry number.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent.

```
171     (%253)   No response received on bid/poll/select, or
                 reply invalid.  (device type:  7, 10, 11,
                 58, 60, or 61)
```

Cause. The selected controller or device did not respond.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
172     (%254)   Reply not proper for protocol; invalid
                 control sequence or invalid data.
                 (device type:  6, 7, 9, 10, 11, or 60)
```

Cause. The selected device responded with an invalid control sequence or invalid data.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
173     (%255)   Maximum allowable NAKs received
                 (transmission error) (device type:  6, 7,
                 10, or 60), invalid MCW on WRITE, or invalid
                 request ID (device type:  11).
```

Cause. The specific meaning for this error is device dependent.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. For a 6520, 6524, or 6530 terminal, check that the power is on. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
174     (%256)   WACK received or CLB frame aborted.
                 (device type:  11)
```

Cause. Either a wait-for-acknowledgment (WACK) sequence was received as the text acknowledgment or a link request occurred while the request was pending.

Effect. The procedure sets the error code and returns without performing the requested operation. Data might be lost.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
175     (%257)   Incorrect alternating ACK received
                 (device type:  7.0, 7.1, 7.2, or 7.3), or
                 command rejected (device type:  11).
```

Cause. Either an incorrect alternating acknowledgment (ACK) was received or a command reject condition was generated.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
176     (%260)   Poll sequence ended with no responder.
                 (device type:  7.3, 7.8, 7.9, or 11.40)
```

Cause. The poll sequence ended, but no message was received in response.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device or application dependent.

```
177     (%261)   Text overrun (insufficient buffer space for
                 data transfer).  (device type:  7, 10, 11,
                 or 60)
```

Cause. The data received on a read exceeds the amount allowed by the read count.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action usually involves increasing the read count; refer to the manual for the specific device for more information. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
178     (%262)   No address list specified.
                 (device type:  7.2, 7.3, 7.8, 7.9, 11.40, or
                 61)
```

Cause. An address list was required for this operation, but none was specified.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent and application dependent.

```
179     (%263)   Application buffer is incorrect
                 (device type:  10 or 61), control request
                 pending, or autopoll active
                 (device type:  11.40).
```

Cause. For an AM3270, TR3271, or X25AM data-communication line, an error was encountered in the application buffer, typically a bad WCC on packet type.

For an EnvoyACP/XF line, the operation could not be performed because a control request was pending or the auto poll feature was active.

For a SNAX line, there was a cryptography failure.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent and application dependent.

```
180     (%264)   Unknown device status received.
                 (device type:  5.3, 6.6 through 6.10, or 10)
```

Cause. An invalid device status was received and could not be translated into a usable error number.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Use the Subsystem Control Facility (SCF) to determine what status was received.

```
181     (%265)   Subdevice expected status information but
                 received data instead. (device type:  10)
```

Cause. Data was sent to the subdevice, but the subdevice expected status information.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
182     (%266)   SNALU access method outbound RU error; see
                 SNALU error code in status area of SNAX
                 header.  (device type:  14.2)
```

Cause. SNAX was unable to transmit an outbound response unit (RU), and the detailed SNAX application logical unit (SNALU) error code is contained in the status area (2 bytes) of the SNAX header.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent.

```
183     (%267)   SNA session has ended.  (device type:  14.2)
```

Cause. The file-system procedure call issued by the primary or secondary logical unit could not be performed because the logical-unit-to-logical-unit session associated with the specified device (*filenum*) no longer exists.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Close, then reopen the process.

```
187     (%273)   Operation returning with no useful data.
```

Cause. CONTROL 26 was previously issued to complete all outstanding requests from this application.

Effect. All pending and active READ, WRITE, and WRITEREAD requests are completed, and no data is transferred.

Recovery. Recovery is application dependent.

```
188     (%274)   Damage to logical flow of events.
```

Cause. CONTROL 26 was previously issued to complete all outstanding requests from this application.

Effect. The state of the session is suspect.

Recovery. Recovery is application dependent.

```
189     (%275)   Response not yet available.
```

Cause. CONTROL 26 was previously issued to complete all outstanding requests from this application.

Effect. The requesting program concludes that the actions initiated by a READ, WRITE, or WRITEREAD request are in progress, and the response that would have been returned is not yet available.

Recovery. Recovery is application dependent. A generic read operation will retrieve the response when the response is available.

```
190     (%276)   Device error; hardware problem.
                 (device type:  any except 2)
```

Cause. A hardware error occurred and persisted through several retries of the operation. A problem exists with either the device or its controller. The error might be caused by a termination condition that was not expected.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent. If this problem persists and a data communications process is involved, obtain a trace and submit it to your service provider.

```
191     (%277)   Device power on (device type:  5), or
                 terminal reset (device type:  6).
```

Cause. The device power was switched off, perhaps due to power failure, then was switched back on during this operation.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent.

```
192     (%300)   Device in diagnose mode; system operator
                 running diagnostics. (device type:  3, 4, 5,
                 or 6)
```

Cause. The system operator was running diagnostic programs on the specified device.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait, then try again.

```
193     (%301)   Invalid or missing microcode file.
                 (device type:  any except 3)
```

Cause. The operating system could not do one of the following:

- Locate the microcode files for a downloadable controller
- Read either of the microcode files because of disk file errors
- Download from the microcode files because they are not formatted properly

Effect. The procedure sets the error code and returns without performing the requested operation. The operator message MICROCODE LOADING FAILURE (console message 100, EMS operator message 100, or DISK operator message 5035) is displayed twice, once for the primary file and once for the backup.

Recovery. See the *Operator Messages Manual* for information on correcting this error.

```
194     (%302)   Device use or mount request rejected by
                 operator. (device type:  4)
```

Cause. A MEDIACOM REJECT TAPEMOUNT command (on G-series systems) was issued to reject a mount request or to request use of a drive for an unlabeled tape.

Effect. The open fails.

Recovery. Informative message only; no corrective action is needed.

```
195     (%303)   Operation requires use of $ZSVR but it is
                 not running; tape operation is not allowed.
                 (device type:  4)
```

Cause. On a system with tape label processing enabled, the named process $ZSVR was not running when a tape operation was requested to open a tape file.

Effect. No tape operation is allowed.

Recovery. Start $ZSVR. For additional information, see the appropriate system operator’s guide.

```
196     (%304)   A tape label record is missing or incorrect.
                 (device type:  4)
```

Cause. The label record on the tape volume was incorrect or missing or a DEFINE attribute value did not match the value for that attribute recorded in the tape label.

Effect. Access to the tape file is denied.

Recovery. Check that the correct tape was mounted. If a mismatch of DEFINE attribute values occurred, modify the DEFINE attribute that did not match the tape label. Use the MEDIACOM INFO TAPELABEL command (on G-series systems) to dump the beginning of the volume group labels on the tape. Correct the label accordingly. For additional information, see the appropriate system operator’s guide.

```
197     (%305)   An SQL error has occurred.
```

Cause. A more detailed error explanation cannot be given because this system does not support SQL objects.

If an Enscribe operation (such as PURGE, FUP ALTER, SETMODE, or FUP SECURE) is attempted on an SQL sensitive file, this error is returned.

Effect. The operation fails.

Recovery. The operation can only be performed if your system uses the NonStop SQL/MP product.

```
198     (%306)   A DEFINE of the given name could not be
                 found.  (device type:  any)
```

Cause. The specified DEFINE name does not exist.

Effect. The request is ignored.

Recovery. Either correct the name or supply a DEFINE of the given name.

```
199     (%307)   The disk file is SAFEGUARD protected.
                 (device type:  3)
```

Cause. An attempt was made to change a security attribute (security vector, license, PROGID, and CLEARONPURGE) for a file protected by Safeguard.

Effect. No security attributes can be established for the file through the file system or through FUP.

Recovery. Use SAFECOM to establish the security attributes.

```
200     (%310)   The device is owned by an alternate port.
                 (device type:  any except 2)
```

Cause. The logical ownership of the device was switched to the other processor to which the device is configured.

Effect. The file system automatically retries the operation.

Recovery. Informative message only; no corrective action is needed.

```
201     (%311)   The current path to the device is down
                 (device type:  any except 0 and 2), an
                 attempt was made to write to a non-existent
                 process (device type:  0), the
                 message-system request was incorrectly
                 formatted, or an error was found in the
                 message system interface.
```

Cause. This error can occur for the following reasons:

- A request sent to a process or device on a remote system by way of the Expand subsystem received this error because the Expand process on the remote system could not deliver the request to the process or device.
- There was an incorrectly formatted message-system request.
- For a process file, either an attempt was made to write to a nonexistent process or a pending WRITE or WRITEREAD was aborted because the server process read the request using READUPDATE but died before it could reply.

- For a device, the current path to the device was down.

Effect. Either the operation never starts, the operation finishes but the path fails before a reply can be made, or the processor fails.

For a process file opened with a sync depth of 0, the primary process fails if the process file is a process pair.

Recovery. For a process file, retry the operation once to establish communication with the backup process. If a second error 201 occurs, no backup exists (both paths are down) and programmatic recovery is impossible.

For disk server processes, open these processes with a sync depth greater than 0.

Then take corrective action appropriate to the device and the application.

```
210     (%322)   Device ownership changed during operation.
                 (device type:  any except 2)
```

Cause. This error is associated with concurrent operations involving more than one unit connected to a multiunit controller.

This error occurs when an operation is in progress in one unit on a multiunit controller and an error is detected during an operation with another unit on the same controller. (The other operation could have been on behalf of this or another application process.)

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is device dependent and application dependent.

```
211     (%323)   The processor performing the operation
                 failed during the operation.
                 (device type:  any)
```

Cause. The processor module controlling the device associated with this file operation failed (path error).

Effect. The procedure sets the error code and returns without performing the requested operation. The file operation itself stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
213     (%325)   Channel data parity error (path error).
                 (device type:  any except 2)
```

Cause. A controller or channel failure occurred (path error).

For a 3106 controller, a command could have arrived at the same instant that the controller was busy processing a seek. However, if the problem is due to 3106 synchronization, this is an informative message only.

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

For the 3106 controller, use the IOTRACE program to examine the I/O trace area of the affected processor.

```
214     (%326)   Channel timeout (path error).
                 (device type:  any except 2)
```

Cause. A controller or channel failure occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
215     (%327)   I/O attempted to absent memory page
                 (hardware path error).
                 (device type:  any except 2)
```

Cause. A hardware failure occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
216     (%330)   Memory breakpoint encountered during this
                 I/O operation.  (device type:  any except 2)
```

Cause. A memory-access breakpoint was encountered during this operation.

Effect. The file operation stopped at some indeterminate point.

Recovery. If the breakpoint was not intentionally set, clear it using Debug. Note which system call was used in the application and/or which system process is affected. If the problem persists, contact your service provider.

```
217     (%331)   Memory parity error during this I/O
                 (hardware path error).
                 (device type:  any except 2)
```

Cause. A hardware failure occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
218     (%332)   Interrupt timeout occurred on a channel, or
                 a controller, modem, or the line between, or
                 lost the modem clock (path error).
                 (device type:  any except 2)
```

Cause. A controller failure, channel failure, line disconnect between controller and modem, or loss of modem clock occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
219     (%333)   Illegal device reconnect (path error).
                 (device type:  any except 2)
```

Cause. An invalid device reconnect occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
220     (%334)   Protect violation; an I/O controller
                 attempted an illegal write.
                 (device type:  any except 2)
```

Cause. An I/O controller tried to write to memory when it should not have done so.

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
221     (%335)   Controller handshake violation (path error).
                 (device type:  any except 2)
```

Cause. A controller or channel failure occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
222     (%336)   Bad channel status from EIO instruction
                 (path error).  (device type:  any except 2)
```

Cause. A controller or channel failure occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
223     (%337)   Bad channel status from IIO instruction
                 (path error).  (device type:  any except 2)
```

Cause. A controller or channel failure occurred (path error).

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
224     (%340)   Controller error (fatal error).
                 (device type:  any)
```

Cause. The controller has a fatal error that was recognized by its resident microcode.

Effect. All paths through the controller are down.

Recovery. Contact your service provider.

```
225     (%341)   No unit assigned or multiple units assigned
                 to the same unit number (path error).
                 (device type:  any except 2)
```

Cause. A path error occurred because no unit was assigned to the unit number or because multiple units were assigned to the same unit number.

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent. If the device is a disk, check the UNITS plugs on the drive connected to the controller.

```
230     (%346)   Processor power failed, then restored.
                 (device type:  any except 2)
```

Cause. The processor power failed and then was restored during this operation (path error). At least one path, and possibly both paths, is operable.

Effect. The file operation stopped at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent.

```
231     (%347)   Controller power failed, then restored.
                 (device type:  any except 2)
```

Cause. The controller power failed and then was restored during this operation (path error). At least one path, and possibly both paths, is operable.

Effect. The file operation stops at some indeterminate point.

Recovery. Corrective action is device dependent and application dependent. Check the log for a channel reset (operator message 77) which can also cause a file-system error 231 to occur (the power-on interrupt causes all controllers on the channel bus to be cleared so they can continue processing).

```
232     (%350)   Access is denied due to error in
                 communication with SMON.
```

Cause. There is a lack of Safeguard resource to support the request; this is an internal error.

Effect. Access is not possible at this time.

Recovery. Wait, then try again. The number of retries is limited.

```
233     (%351)   Error in call to SERVERCLASS_SEND_.
```

Cause. An error occurred in a call to SERVERCLASS_SEND_.

Effect. The call is not accepted.

Recovery. Correct the error in the call and try again.

```
240     (%360)   Network line handler error; operation not
                 started. EXPAND performed an ownership
                 switch.  (device type:  63)
```

Cause. The Expand process performed an ownership switch.

Effect. All traffic not yet sent to the remote system, and traffic that belongs to the affected Expand process is terminated. The application should not see this error.

Recovery. The file system automatically retries. For additional information, see the *Expand Network Management Guide*.

```
241     (%361)   Network error; operation not started.
                 (device type:  63)
```

Cause. The network control process ($NCP) caused the Expand process to terminate its unsent traffic for one of the following reasons:

- A connection to a remote system was established or reestablished over the same path or a different path.
- The network control process came up (for example, after a crash or a takeover by the backup).
- There was no longer a path to the remote system (console message 43, LDEV *ldev* NET: CONNECTION LOST TO SYS *nnn*, is generated by the NCP).

Or, the Expand process encountered one of the following situations:

- The Layer 4 protocol detected a protocol error.
- An I/O power on occurred but all lines used by the Expand process were attached to the controller for which the IOPON occurred.
- A processor power on occurred.
- The Expand process received a request from a system for which it is not the current path.
- An Expand process received a request to forward a message, but all its lines are down.

Effect. The procedure sets the error code and returns without performing the requested operation. The application should not see this error.

Recovery. The file system retries automatically. For additional information, see the *Expand Network Management Guide*.

```
246     (%366)   External cluster bypass error; operation
                 aborted.  (device type:  63, subtype 3)
```

Cause. A message from one cluster to another bypassed the Expand process, went directly to its destination, and terminated due to a disconnect or resynchronization of the bus protocol between clusters.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Retry the operation if possible. For additional information, see the *Expand Network Management Guide*.

```
248     (%370)   Network line handler error; operation
                 aborted.  (device type:  63)
```

Cause. If this error was returned to the application, the Expand process performed an ownership switch. The request, sent to its remote destination, is aborted because the operating system does not know the state of the request.

If the error was returned in Expand or X25AM operator message 45, or in SNAX operator message 23 or 24 (line not ready), the network process was unable to establish level-2 communications or all level-2 retries were exhausted. Possible reasons for this error are:

1. The other system was down
2. There was an incorrect NEXTSYS parameter in the configuration
3. A correctly configured Expand line was initially ready
4. There was garbled data or no data in or out

Effect. The procedure sets the error code and returns. The request to perform the operation is sent but it might not have been performed.

Recovery. If the error is returned to the application, determine whether the operation should be retried. If so, retry the request.

If the error is returned in an operator message, recovery is as follows for the corresponding cause:

1. Select an alternate path if one exists. The state of the other system can be determined by issuing an SCF INFO PROCESS $NCP, NETMAP command.
2. The Expand process is not usable until the correct NEXTSYS parameter is supplied. Either correct the NEXTSYS parameter in the SYSGEN file and perform another SYSGEN or use the Subsystem Control Facility (SCF) to alter the NEXTSYS parameter and then bring the line up using SCF.

3. The line normally exhibits error 248 on the operator console when the line is initially brought up. The message should be followed by another message announcing that the line is up.
4. By performing successive SCF STATS commands (at one minute intervals) and observing the U-FRAME counts, you can determine whether data is being transmitted and received. The FCS error count indicates that garbled data was received.

   If the SCF STATS commands do not indicate both send and receive data, place the local modem in analog loopback and observe the U-FRAME counts again. If both send and receive counts are incrementing, the local controller and modem are working properly. If both systems check out, first check the lines by using the modem self test, then observe the U-FRAME counts with the remote modem in digital loopback. If all tests indicate data is being transmitted and received, then a trace at both ends of the line should indicate the cause of the not-ready condition.

If this problem persists, your service provider needs to run traces or perform other corrective action. Refer to the *Expand Network Management Guide* for further information.

```
249     (%371)   Network error; operation aborted.
                 (device type:  63)
```

Cause. The network control process ($NCP) caused the Expand process to terminate its traffic (already sent to its remote destination) for one of the following reasons:

- A connection to a remote system was established or reestablished over the same path or a different path.
- The network control process came up (for example, after a crash or a takeover by the backup).
- There was no longer a path to the remote system (operator message 43, LDEV *`ldev`* NET: CONNECTION LOST TO SYS *`nnn`*, is generated by the network control process).

Or, the Expand process encountered one of the following situations:

- The level-4 protocol detected a protocol error.
- An I/O power on occurred but all lines used by the Expand process were attached to the controller for which the IOPON occurred.
- A processor power on occurred.
- The Expand process received a request from a system for which it is not the current path.
- A Expand process received a request to forward a message, but all its lines were down.

Effect. The procedure sets the error code and returns. The request to perform the operation is sent but it might not have been performed.

Recovery. Corrective action is device dependent and application dependent. If this problem persists, your service provider needs to run traces or perform other corrective action. For additional information, see the *Expand Network Management Guide*.

```
250     (%372)   All paths to the system are down.
                 (device type:  63)
```

Cause. The referenced system in the network is down, or it is not currently connected to the system on which the requested process is running (path error).

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent. If this problem persists, your service provider needs to run traces or perform other corrective action.

```
251     (%373)   Network protocol error (path error).
                 (device type:  any except 2)
```

Cause. A network protocol error occurred (path error).

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Corrective action is application dependent. If sync depth is greater than 0, keep trying until successful or until a different error is returned.

If this problem persists, your service provider needs to run traces or perform other corrective action.

```
252     (%374)   Required EXPAND class is not available.
```

Cause. The Expand class of transmission (such as FOX) is busy or is not available.

Effect. The message is not sent.

Recovery. Check for the availability of the required class and retransmit the message. If this problem persists, your service provider needs to run traces or other corrective action.

```
255     (%377)   Net line handler flooded; too many
                 interrupts.
```

Cause. There were too many ownership errors. The network process was flooded.

Effect. All requests are aborted, and all lines go into the not-ready state. If all possible combinations of ownership switches are unsuccessful, the lines go down.

Recovery. Check the operator messages to determine the cause of the ownership errors. Correct the cause of the problem as required. If this problem persists, your service provider needs to run traces or other corrective action.

```
300                (%454
through             through  Reserved for application
processes.
511 (except 538)    %777)
```

These file-system error numbers are reserved for use by application processes.

```
538     (%)      Extent size is greater than 65535 pages.
```

Cause. The CHECK^FILE procedure could not return the primary or secondary extent value because the extent size is greater than 65,535 pages.

Effect. The operation is not executed.

Recovery. This is an informational message only; no corrective action is needed.

```
541     (%1035)  A data structure version is incompatible
                 with the requested operation.
```

Cause. The version of a data structure does not support a requested operation.

Effect. The requested operation is not performed.

Recovery. Contact your service provider.

```
549     (%1045)  Blockmode is not currently allowed on this
                 terminal.
```

Cause. The application attempted to perform a SETMODE 8 (with *param1* greater than or equal to 1), but block mode use was not allowed because the terminal was being used as a console device.

Effect. The operation failed.

Recovery. If a block mode application needs to use the device, direct the console to another terminal; the operator process will stop using the device and allow it to be used in block mode.

```
550     (%1046)  File operation attempted at illegal
                 position.
```

Cause. The current setting of the file's positioning information is not valid for the attempted operation. For example, an I/O was attempted under the large transfer SETMODE while the current position was not a multiple of 2048; usually this condition is caused when the last read before an end-of-file returned a number of bytes that is not a multiple of 2048.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Determine the reason for the incorrect position (for example, a bad POSITION call or an unexpected read transfer count) and correct the problem.

```
551     (%1047)  Duplicate exists for insertion-ordered
                 alternate key.
```

Cause. More than one insertion-ordered alternate key record exists.

Effect. Depending on the setting of a bit in the CREATE procedure or an item code in the FILE_CREATELIST_ procedure, duplicate records with insertion-ordered alternate keys are ordered by the value of the primary record key field or by the sequence in which those records were inserted into the alternate key file.

Recovery. Informative message only; no corrective action is needed.

```
560     (%1060)  The function cannot be performed because one
                 of the processes involved cannot recognize a
                 PIN greater than 255.
```

Cause. The function cannot be performed because one of the processes involved cannot run with a process identification number (PIN) greater than 255. A process created by the D-series procedure PROCESS_LAUNCH_ or PROCESS_CREATE_ can request to be run at a high PIN (greater than 255), but a process created by the C-series procedure NEWPROCESS or NEWPROCESSNOWAIT cannot run at a high PIN.

Effect. The operation failed.

Recovery. Any of the following can be performed to recover:

- One or more of the processes involved must be rerun with low PINs (PINs less than 255).
- One or more of the processes involved must be changed to allow for high-PIN requestors by setting HIGHPINREQUESTORS ON. A process that ran at a low PIN continues to run at a low PIN but can now be opened by high-PIN processes. Take this action only if the application can be opened by a high-PIN process.
- One or more of the processes involved must be changed to support high PINs.

```
561     (%1061)  The item code in a list is not recognized.
```

Cause. The item code in a list was not recognized.

Effect. Either FILE_CREATELIST_ failed or FILE_GETINFOLIST_ or FILE_GETINFOLISTBYNAME_ stopped at an item in error.

Recovery. The item code in error should be corrected.

```
563     (%1063)  The size of an output buffer was too small.
```

Cause. The size of an output buffer was too small to hold the data. Error 563 is an operating system error and should not be confused with error 177, which is a data communications error.

Effect. The operation failed.

Recovery. Increase the size of the buffer.

```
564     (%1064)  The operation is not supported for this file
                 type.
```

Cause. The operation specified is not permitted on this file type.

Effect. The operation failed. Error 564 is returned rather than error 2 when an unqualified failure occurs.

Recovery. Determine why an invalid operation was requested and correct the request.

```
565     (%1065)  A malformed request was denied.
```

Cause. The server that received the message found the request malformed or not appropriate under current conditions.

Effect. The operation failed.

Recovery. The problem is probably on the requester side. Verify the conditions of the requester; if they are satisfied, contact your service provider.

```
566     (%1066)  This reply is malformed or not appropriate.
```

Cause. The requester found the reply from the server was malformed or not appropriate under current conditions. The error is returned to the requester, not to the reply procedure, because the problem is detected in the requester's reply handling (such as AWAITIO[X]).

Effect. The operation failed.

Recovery. The problem is probably on the server side. Verify the conditions of the server; if they are satisfied, contact your service provider.

```
567     (%1067)  The define used is incompatible for use with
                 target system's TOS version.
```

Cause. A CLASS of DEFINE is being used in a context where it is not supported by the operating system level.

Effect. The operation failed.

Recovery. Increase the operating system level of the target system or use a supported CLASS of DEFINE.

```
573     (%1075)  The requested process handle cannot be
                 returned.
```

Cause. A process control operation was attempted using a process handle of a system process (IOP).

Effect. The operation failed.

Recovery. Only attempt the operation on user processes.

```
578   (%1102)    The block size specified is too large.
```

Cause. The specified block size is too large.

Effect. The requested operation is not performed.

Recovery. Reduce the block size specified in the file creation request to an acceptable value. For format 1 files, the largest allowed value is 4096 bytes.

```
579   (%1103)    The record size specified is too large for
                 the given block size, file type and format.
```

Cause. The specified record size is too large for the given block size, file type and format.

Effect. The requested operation is not performed.

Recovery. Increase the block size or reduce the record size specified in the file creation request. The largest supported record size in a format 1 file is slightly less than in a format 2 file for a given block size.

```
580  (%1104)      An open failed because the file was oversize
                 and the opener did not specify use of 64-bit
                 primary keys.
```

Cause. An open failed because the file was a format 2 file and the opener did not specify use of 64-bit primary keys. An attempt was made to open a relative file, an entry-sequenced file, or an unstructured file with a potential size of over 4 gigabytes without use of a special indicator specifying that the program was using 64-bit keys rather than 32-bit keys.

Effect. The requested operation is not performed.

Recovery. Update the application to use 64-bit keys. This involves changing the use of 32-bit key values to 64-bit values, replacing calls to 32-bit procedures (such as POSITION) with equivalent procedure calls (for example, FILE_SETPOSITION_), and setting a 64-bit indicator in the FILE_OPEN_ procedure call. If the file does not contain and does not need to contain 4 GB of data, an alternative is to reduce the maximum file size (for example, reducing the maximum number of extents).

```
581  (%1105)     An operation involving 32-bit primary keys
                 was attempted on an open which specified use
                 of 64-bit keys.
```

Cause. An operation involving 32-bit primary keys was attempted on an open which specified use of 64-bit keys.

Effect. The requested operation is not performed.

Recovery. Correct the application by replacing the 32-bit procedure call that caused the error with an equivalent procedure call that accepts the use of 64-bit keys. Alternatively, turn off the FILE_OPEN_ procedure indicator that specifies the use of 64-bit keys. The second option will not allow the application to access non-key-sequenced files that are 4 GB or larger.

```
582  (%1106)     Alternate key information could not be
                 returned because it cannot be expressed in
                 the superseded format of the parameter.
```

Cause. Alternate key information could not be returned, because it cannot be expressed in the superseded format of the parameter. For instance, if an alternate key field has an offset greater than 4095, this error will be returned if the alternate key information is requested using the FILEINQUIRE procedure, because the format from this procedure allows only 12 bits for an offset.

Effect. The requested operation is not performed.

Recovery. Update the application to use an interface supporting the full range of alternate key information, for instance by using FILE_GETINFOLIST_ item 106.

```
583  (%1107)     The extent size specified is too large or
                 the maxextents limit is too large.
```

Cause. The specified extent size is too large or a *maxextents* value bigger than 16 is specified for a non-key sequenced partitioned file. Usually this error occurs because an extent size of more than 65,535 pages is specified for a format 1 file. The extent size in question might be the primary or secondary extent size of a file, or it might be any of the extent sizes specified in the partition description.

Effect. The requested operation is not performed.

Recovery. For the extent size problem, reduce the extent size or arrange the conversion of the file into format 2, which can handle larger extent sizes than format 1 files can. For the partitioned *`maxextents`* problem, avoid using a *`maxextents`* value greater than 16.

```
584  (%1110)     The operation could not be performed because
                 a software component does not support
                 format 2 disk files.
```

Cause. The operation could not be performed because a software component does not support format 2 disk files. The component may be local to the caller or may be on a remote system if a remote object is being manipulated.

Effect. The requested operation is not performed.

Recovery. Check the software versions on the local system and, if applicable, the remote system to determine which component does not provide support for format 2 files. Some subsystems, such as optical disk, might not have a version available which can support format 2 files. There might also be down-level version disk volume references in partition and alternate key descriptions. Upgrade the necessary systems, or move the objects to a systems supporting format 2 files.

```
590     (%1116)  The parameter value is invalid or
                 inconsistent with another.
```

Cause. A parameter value is not valid, or two parameter values are not compatible. Error 590 is returned when no specific error applies to the problem.

Effect. The operation failed.

Recovery. The value of the parameter must be an acceptable value. Correct the problem and try again.

```
593     (%1121)  The request was cancelled.
```

Cause. The request was abandoned.

Effect. The operation failed.

Recovery. Informational message only; no corrective action is needed.

```
594     (%1122)  A DSM/TC error was returned to $ZSVR. Refer
                 to the EMS log for detailed information.
```

Cause. DSM/TC returned a fatal error to $ZSVR.

Effect. The tape request failed.

Recovery. See the EMS log for details about the DSM/TC error. Then see the *DSM/Tape Catalog Messages Manual* for recovery actions.

```
595     (%1123)  A ZSSI error was returned to $ZSVR.  Refer
                 to the EMS log for detailed information.
```

Cause. ZSSI returned a fatal error to $ZSVR.

Effect. The tape request failed.

Recovery. See the EMS log for details about the ZSSI error. Then see the *DSM/Tape Catalog Messages Manual* for recovery actions.

```
597     (%1125)  A required item is missing from an item
                 list.
```

Cause. An item code that should have been specified in an item list was not found.

Effect. The operation failed.

Recovery. For each item in the item list, check that all of its required items are present and are in the correct order.

```
632     (%1170)  Not enough stack space to complete request.
```

Cause. A procedure is called but less than the required amount of data stack space is available. Some inadequate stack situations can cause this error; others cause a trap instead.

Effect. The last error is set to this error number, and the requested operation is not completed.

Recovery. Increase the number of stack pages available or reduce the amount of stack space used.

```
634     (%1172)  A logical device number exceeded 16 bits.
```

Cause. A logical device number too large to fit in a 16-bit field (greater than 65535) was used.

Effect. The requested operation is not completed.

Recovery. Reduce the number of bits in the logical device number or, if the error was returned from a procedure call in a user application, convert the procedure call to use a new equivalent procedure that supports 32-bit logical device numbers.

```
635     (%1173)  A volume cannot be accessed because the
                 other side is locked.
```

Cause. This error is returned under either of the following conditions:

- An attempt was made to lock an optical disk volume, but either both drives were not up or one volume was already locked.
- An attempt was made to access an optical disk volume, but the volume on the other side of the cartridge was locked.

Effect. The optical disk volume either cannot be locked or cannot be accessed.

Recovery. Retry the locking operation after seeing that both optical disk drives are up and both volumes are unlocked, or retry the access operation after unlocking the volume on the other side of the cartridge.

```
638     (%1176)  Process cannot be stopped until process
                 returns to stopmode 1.
```

Cause. A stop request passes the security checks but the process is running at stopmode 2.

Effect. The stop request is queued until the process to be stopped reduces its stopmode.

Recovery. Informative message only; no corrective action is needed.

```
639     (%1177)  Process cannot be stopped until process goes
                 to stopmode 0.
```

Cause. A stop request does not pass the security checks and the process is running at stopmode 1 or 2.

Effect. The stop request is queued until the process to be stopped reduces its stopmode to 0.

Recovery. Informative message only; no corrective action is needed.

```
640     (%1200)  DEBUGNOW is required instead of DEBUG for a
                 privileged process.
```

Cause. DEBUGNOW is required instead of DEBUG for a privileged process.

Effect. Debug mode will not be entered.

Recovery. Use DEBUGNOW instead of DEBUG

```
700     (%1274)  FEWRONGRECOVSEQNUM - The sequence number of
                 the message received by the disk process
                 from a recovery process does not match.
```

Cause. DP2 received an incorrect message sequence number.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
701     (%1275)  FEDEVICEDOWNFORTRANSACTIONS - The disk
                 process received a message from a recovery
                 process that requires that the disk process
                 be in the UP state; it is currently in the
                 DOWN state.
```

Cause. Volume is in the `DOWN` state.

Effect. The operation fails. TMF generates an EMS event that provides information regarding the error.

Recovery. The system administrator must change the volume to the `UP` state.

```
702     (%1276)  FEUNSUPPORTEDOP - The disk process received
                 an erroneous message from a recovery
                 process. The message requested a physical
                 REDO and also requested that audit be
                 generated.
```

Cause. The request is not understood by DP2.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
703     (%1277)  FEAUDITTOONEW - The disk process encountered
                 a Creation Volume Sequence Number in an
                 audit record sent by a recovery process that
                 is more recent than the CRVSN of the File
                 Label.
```

Cause. The label stored by DP2 is older than the label described in the audit record.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
704     (%1278)  FEPVSNMISMATCH - The disk process
                 encountered a Previous Volume Sequence
                 Number in an audit record sent by a recovery
                 process that does not match the VSN of the
                 data block on disk.
```

Cause. The label stored by DP2 is older than the label described in the audit record.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
705     (%1279)  FENOTANADP - Generated by the disk process
                 when it receives a TMF Auditing Disk Process
                 request type message and the disk process is
                 not an ADP volume.
```

Cause. The disk process is not an Auditing Disk Process.

Effect. TMF does not work.

Recovery. Contact your HP service provider.

```
706     (%1280)  FEINVALFORADP - Generated by the disk
                 process when it receives a request message
                 that is inappropriate for an Auditing Disk
                 Process.
```

Cause. The request is inappropriate for Auditing Disk Process.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
707     (%1281)  FEINVALIDDPNAMETIMESTAMP - Generated by the
                 disk process when the Disk Process Name
                 Stamp(DPNameTimeStamp)in the message sent by
                 a recovery process does not match the
                 current DPNameTimeStamp of the disk.
```

Cause. When the `DPNameTimeStamp` of the request does not match the `DPNameTimeStamp` of the disk process.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
708     (%1282)  FEFILEUNDONEEDED - The disk process
                 encountered a File Label that had its
                 UndoNeeded flag set when a recovery request
                 specified that the UndoNeeded flag must not
                 be set.
```

Cause. The request specifies that the file is currently marked `UndoNeeded` but does not have its `UndoNeeded` flag set.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
709     (%1283)  FEFILEREDONEEDED - The disk process
                 encountered a File Label that had its
                 RedoNeeded flag set when a recovery request
                 specified that the RedoNeeded flag must not
                 be set.
```

Cause. The request specifies that the file is currently marked `RedoNeeded` but does not have its `RedoNeeded` flag set.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
710     (%1284)  FENULLOP - No further work needs to be done
                 to process the current portion of the
                 current request OPENTMF errors.
```

Cause. The current audit record does not need to be applied.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
711     (%1285)  FEINVALAUDITREC - The disk process received
                 a corrupt audit record in a message from a
                 recovery process.
```

Cause. The audit record is corrupted.

Effect. The operation fails.

Recovery. Contact your HP service provider.

```
712     (%1310)  FERMALREADYREGISTERED
```

Cause. An attempt was made to register a recoverable resource manager using the TMF_REC_RM_CREATE_ procedure, but the name was already registered.

Effect. No change is made to the resource manager directory.

Recovery. Change the name to be registered and try again if needed.

```
713     (%1311)  FERMOUTSTANDINGTRANS
```

Cause. An attempt was made to remove a recoverable resource manager from the directory using the TM_REC_RM_REMOVE_ procedure, but the resource manager still has unresolved transactions outstanding.

Effect. The resource manager directory is not changed.

Recovery. Resolve the transactions and then retry the remove operation. The transactions can be resolved through the system management interface (TMF-COM/TMFSERVE) or the resource manager can be opened and the transactions can be resolved by communicating with the appropriate foreign transaction manager.

```
714      (%1312)   FEINVALIDPROTOCOL
```

Cause. A call to TMF_EXPORT_ or TMF_IMPORT_ specified an invalid combination of protocols and options for the transaction branch.

Effect. The operation, either export or import, did not succeed.

Repeat the operation with a valid combination of protocol and options.

```
715     (%1313)    FEINVALIDTXHANDLE
```

Cause. An invalid transaction handle was specified. The handle could be completely invalid, or the type of handle could be invalid for the operation specified, such as trying to do a TMF_SETTXHANDLE_ using the transaction handle returned from TMF_EXPORT_ for a non pre-prepare branch.

Effect. The requested operation is not performed.

Recovery. Retry the operation with a valid transaction handle, if possible.

```
716     (%1314)    FETXSUSPENDREJECTED
```

Cause. An attempt was made to suspend a transaction (TMF_SUSPEND_), but the caller is not the beginner nor the importer nor the “resumer” of the transaction.

Effect. The transaction is not suspended for the process.

Recovery. No recovery is necessary. However, if the process expected the operation to complete successfully, the problem should be investigated.

```
717     (%1315)    FETXNOTSUSPENDED
```

Cause. An attempt was made to resume a transaction (TMF_RESUME_), but the specified transaction is not currently suspended.

Effect. The transaction is not resumed by the calling process.

Recovery. Determine if the transaction should be suspended and why it is not and repair the application as appropriate.

```
718     (%1316)    FEINVALIDSIGNAL
```

Cause. An invalid signal value was found. Either an invalid value was specified to TMF_WRITE_SIGNAL_ or an invalid buffer was specified to TMF_INTERPRET_SIGNAL_.

Effect. The operation is not performed.

Recovery. Determine why the invalid signal value was used or why the TMF subsystem thinks the buffer is invalid.

```
719    (%1317)    FEDATASIZEEXCEEDED
```

Cause. An attempt was made to write branch or resource manager data, but the amount of data specified would cause the allowable limits on the data to be exceeded.

Effect. In the case of branch data (TMF_WRITE_TX_DATA_) the transaction is aborted. In the case of resource manager data (TMF_WRITE_RM_DATA_), the operation fails and the resource manager data is not updated.

Recovery. Fix the application to not generate more than the allowable amount of branch or resource manager data. Branch data is limited to 1.5K (1024) bytes per branch, or a total of 12K (12288) bytes per transaction. Resource manager data is limited to 1K (1024) bytes per resource manager.

```
721    (%1321)    FEBEGINTXNOTCOMPLETED
```

Cause. The TMF_RESUME_ procedure was called before the BEGINTRANSACTION processing for the specified transaction is complete. This could occur because the BEGINTRANSACTION procedure does not wait for the processing to complete before returning to the caller and the caller could subsequently call TMF_SUSPEND_ and send the tid off to another process to resume before the processing is complete.

Effect. The TMF_RESUME_ fails.

Recovery. Delay for a few seconds and try the TMF_RESUME_ again.

```
722    (%1322)    FENOMORERMCBS
```

Cause. An attempt was made to open a resource manager file, but the gateway process' local CPU has run out of available resource manager control blocks.

Effect. The resource manager open fails.

Recovery. The number of resource manager control blocks allocated per CPU can be configured through the subsystem management interface. If this error is consistently being returned, that value can be increased. Another possible recovery is to distribute the gateway processes across other CPUs where the limit has not been reached.

```
723    (%1323)    FENOMOREBCBS
```

Cause. An attempt was made to export or import a transaction branch, but the limit of the number of branch control blocks per CPU has been exceeded.

Effect. The operation fails.

Recovery. The number of branches per resource manager can be configured through the subsystem management interface. If this error is consistently being returned, that value should be increased. Another possible recovery is to add more resource managers to the directory and distribute the load of transaction branches across them.

```
724    (%1324)   FENOTNOWAITTFILE
```

Cause. An attempt is made to export a transaction branch with the TFILE not opened in a nowaited manner.

Effect. The Export will fail.

Recovery. Open the TFILE in nowaited manner and then export the transaction branch.

```
725    (%1325)   FEIMPORTINVALOP
```

Cause. An attempt is made to call ENDTRANSACTION on an imported transaction branch.

Effect. The ENDTRANSACTION call fails.

Recovery. As TMF did not start an imported transaction branch, TMF cannot issue ENDTRANSACTION on this transaction.

```
727   (%1327)    FETOOMANYRECRMS
```

Cause. An attempt was made to create a recoverable resource manager, but the maximum allowable recoverable resource managers for the system is exceeded.

Effect. The operation fails.

Recovery. Use the TMFCOM command ALTER BEGINTRANS to increase the total allowable recoverable resource managers in a system. The maximum allowable value is 16384.

```
728    (%1328)   FESETTXHANDLEINVALOP
```

Cause. An attempt was made to set the current transaction of a process by invoking TMF_SETTXHANDLE_ passing in the TxHandle obtained by exporting a transaction branch to a volatile resource manager with pre-prepare option, but the exported transaction branch is prepared.

Effect. The operation fails.

Recovery. If the transaction is still active, and more work on behalf of that transaction must be done by that volatile resource manager, then export another transaction branch.

```
729    (%1331)   FEBRANCHISPREPARED
```

Cause. An attempt was made to write branch data to a prepared exported or imported transaction branch.

Effect. The operation fails.

Recovery. The branch data of a prepared transaction branch cannot be updated.

```
730    (%1332)   FEJOINSOUTSTANDING
```

Cause. There are outstanding TMF_JOIN_s for the process. This error occurs during ENDTRANSACTION processing.

Effect. The operation fails.

Recovery. Call TMF_SUSPEND_ for each outstanding join before calling ENDTRANSACTION.

```
731   (%1333)    FEALREADYJOINED
```

Cause. Transaction has already been joined by this process.

Effect. The operation fails.

Recovery. A process can only have one join outstanding at any time.

```
732    (%1334)   FEALREADYRESUMED
```

Cause. Transaction has already been resumed by this process.

Effect. The operation fails.

Recovery. A process can only have one resume outstanding at any time.

```
733    (%1335)   FEBRANCHISFAILED
```

Cause. The transaction branch has already failed. This error occurs when branch data is updated while the transaction is being aborted.

Effect. The operation fails.

Recovery. Correct the program logic.

```
734    (%1336)   [FEFILENOTRECOVERABLE] DP2 cannot recover
                 the file because audit record cannot be
                 applied.
```

Cause. File recovery cannot be applied to this file because the SQL/MX software version and object version do not match.

Effect. The file recovery operation fails.

Recovery. Contact your HP service provider.

```
735    (%1337)   [FEZLTCOMMITHOLD] DP2 cannot guarantee zero
                 lost transactions and has activated commit
                 hold.
```

Cause. The remote mirror disk is not available, or COMMITHOLDMODE is ON for the volume.

Effect. The operation fails.

Recovery. TMF will activate CommitHold for the volume.

```
758    (%1366)   Unable to allocate space from the control
                 block pool or trying to open too many files
                 or partitions on a volume.(device type: 3)
```

Cause. No space is available in the control block pool to allocate a new control block, or a file or partition open request exceeds the maximum number of opens per volume.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait for a few minutes, and then retry. Check the system for processes that are requesting too many file or partition opens. If the problem persists, contact your service provider.

```
759    (%1367)   Unable to allocate space from audit
                 checkpoint pool. (device type:3)
```

Cause. No space is available in the audit checkpoint pool to allocate a new buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait for a few minutes, and then retry. If the problem persists, contact your service provider.

```
760    (%1370)   Unable to allocate space for a cache
                 buffer.(device type: 3)
```

Cause. No space is available for a new cache buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. This is a Data Access Manager internal error. Contact your service provider.

```
761    (%1371)   Unable to allocate space from the work
                 pool.(device type: 3)
```

Cause. No space is available in the work pool to allocate a new buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait for a few minutes, and then retry. If the problem persists, contact your service provider.

```
762    (%1372)   Unable to allocate space from the free space
                 table.(device type: 3)
```

Cause. No space is available in the disk free space table.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Use the DCOM utility to recover free space on the disk. Then rebuild the free space table by issuing the SCF command CONTROL DISK *$disk-name*, REBUILDDFS. If the problem persists, contact your service provider.

```
763    (%1373)   Unable to allocate space from the local
                 pool.(device type: 3)
```

Cause. No space is available in the local pool to allocate a new buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. This is a Data Access Manager internal error. Contact your service provider.

```
764    (%1374)   Unable to allocate space from the revive
                 pool.(device type: 3)
```

Cause. No space is available in the revive pool to allocate a new buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait for a few minutes, and then retry. If the problem persists, contact your service provider.

```
765    (%1375)   Unable to allocate space from the SQL data
                 area pool (MX buffer space).(device type: 3)
```

Cause. No space is available in the SQL data area pool to allocate a new buffer or reuse an existing buffer. There are too many SQL/MX session requests for buffer space.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Reduce the number of concurrent SQL/MX sessions for the disk volume. If the problem persists, consider increasing the SQLMXBuffer size by issuing the SCF command ALTER DISK *$disk-name*, SQLMXBUFFER *size in MB*. If the problem persists, contact your service provider.

△ **Caution.** Increasing the SQL/MX buffer space reduces the memory available for cache. This can degrade performance.

```
766    (%1376)   The maximum number of opens for a single
                 file or partition has been reached.(device
                 type: 3)
```

Cause. A file or partition open request exceeds the maximum number of opens for a single file or partition on a volume.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait for a few minutes, and then retry. Check the system for processes that are requesting too many opens of a single file or partition. If the problem persists, contact your service provider.

```
899    (%1603)   FENOSWITCH - An attempt to switch CPUs with
                 a PUP or SCF command failed.
```

Cause. The primary or backup DP2 process is not running.

Effect. The operation fails.

Recovery. Ensure that both the primary and backup DP2 processes are running.

```
1091    (%2103)  The file or table cannot be purged until the
                 NOPURGEUNTIL date.
```

Cause. An attempt was made to purge a file for which the specified expiration date has not been reached. The expiration date is set by using item code 57 of the FILE_CREATELIST_ procedure or by using the NOPURGEUNTIL option of the FUP ALTER command.

Effect. On a purge operation, if the NOPURGEUNTIL option is specified, the expiration date (a four-word GMT timestamp) is checked against the current time. If the timestamp is less than the current time, the file is not purged. All Enscribe files existing before C10 are assumed to have a zero expiration date.

Recovery. The expiration date of a file can be set or changed with a call to the C-series ALTER procedure or a call to the D-series FILE_ALTERLIST_ procedure.

```
1163    (%2213)  Illegal operation attempted on a file having
                 a system reserved filename.
```

Cause. An attempt was made to refer to a file that has either a reserved name or cannot be used in the current context. For example, this error is returned when a Guardian internal name for an OSS file is given as the program file in the PROCESS_CREATE_ procedure.

Effect. The operation failed.

Recovery. Correct the error in the call and try again.

```
3502    (%6656)     FEREQUESTALLOCATIONFAILURE
```

Cause. Enough storage was not available at the server to process the request because of the demands of other activities at the time when the request was received.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Wait till the server finishes processing other activities and retry the operation. Contact HP support in case of repeated failure.

```
4000       (%7640          Open System Services Error
through     through
4999        %11607)
```

Cause. A request was made of Open System Services that could not be completed.

Effect. The effect depends on the specific error returned.

Recovery. The recovery action depends on the specific error returned. See Section 22, OSS Error Information for information on how to find the meaning of an OSS error.

```
5001    (%11611) Request rejected: requestor executing on a
                 non-SMF system.  (device type:  3.36, 25.0,
                 or 52.0)
```

Cause. NonStop Storage Management Foundation (SMF) received a request it could not process. The requestor is most likely at a version level earlier than D40.00.

Effect. The requested operation is not performed.

Recovery. The requestor’s system must be upgraded before the request can be processed.

```
5002    (%11612) Creation of logical file rejected pending
                 deletion of volume from storage pool.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. A request to create a logical file was rejected because the target physical volume was being deleted from a storage pool.

Effect. The requested operation is not performed.

Recovery. Create the logical file on a physical volume still associated with a storage pool.

```
5007    (%11617) SMF access violation:  insufficient
                 SMF-privilege to access file.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. An attempt was made to purge or alter an SMF catalog file by some means other than PUP or SMFIXUP.

Effect. The requested operation is not performed.

Recovery. Use PUP or SMFIXUP to operate on SMF catalog files.

```
5010    (%11622) An error occurred during SMF lock manager
                 processing.  (device type:  3.36, 25.0, or
                 52.0)
```

Cause. The NonStop Storage Management Foundation (SMF) internal lock manager encountered an error (for example, a lock could not be obtained because of a problem with a particular file).

Effect. The requested operation is not performed.

Recovery. Appropriate recovery varies according to the specific cause of the error. In some cases you can wait and retry the request. The event log should contain detailed information that can help you determine the appropriate action.

```
5011    (%11623) An error occurred during access of the ANT
                 table.  (device type:  3.36)
```

Cause. A file-system request (create, open, purge, rename, alter, or information inquiry) was made that required a read and/or update of the NonStop Storage Management Foundation (SMF) ANT file, which stores the logical file names and their mappings. A file-system error occurred during the read or update of the ANT file.

Effect. The requested operation is not performed.

Recovery. Appropriate recovery varies according to the file-system error that occurred during access of the ANT file. Refer to the event log to determine the file-system error and take appropriate action.

```
5012    (%11624) An error occurred during access of the
                 PENDOPS table.  (device type:  3.36, 25.0,
                 or 52.0)
```

Cause. A file-system request (create, purge, rename, or alter) was made, or a file recovery operation was occurring, that required access to the NonStop Storage Management Foundation (SMF) PENDOPS (pending operations) file. A file-system error occurred during access of the PENDOPS file.

Effect. The requested operation is not performed.

Recovery. Appropriate recovery varies according to the file-system error that occurred during access of the PENDOPS file. Refer to the event log to determine the file-system error and take appropriate action.

```
5013    (%11625) The name range for a physical volume is
                 exhausted.  (device type:  3.36)
```

Cause. A request was made to create a new NonStop Storage Management Foundation (SMF) logical file, but there are no file names available on the physical volume. In most cases, file names are not available because the virtual disk process has lost communication with the pool process.

This error can occur either when you specify a physical volume or when you do not specify a physical volume and this volume is the last one to be tried. In the latter case, other volumes might have been tried and rejected either because they returned errors or because they were unsuitable (for example, because they did not match the file's criteria for auditing or mirroring).

Effect. The requested operation is not performed.

Recovery. If you specified a physical volume, retry the request either without specifying a physical volume or specifying a different physical volume. If you did not specify a physical volume, start the pool process (if it's not running), wait, and retry the request.

```
5014    (%11626) No physical volumes are available.
                 (device type:  3.36)
```

Cause. A request was made to create a new NonStop Storage Management Foundation (SMF) logical file but there are no physical volumes available for one of the following reasons:

- Either there are no physical volumes in the storage pool that the virtual disk process is associated with, or none of the physical volumes in the pool is in the UP state, or none of the physical volumes in the pool matches the pool's criteria (for example, a pool might require that physical volumes be mirrored, but currently all the physical volume mirrors are in the DOWN state).
- None of the physical volumes in the storage pool matches the criteria specified for the file. For example, the file might be audited but none of the physical volumes in the pool are currently able to generate audit.
- You specified a physical volume on which to create the file, but that physical volume either is not part of the pool that the virtual disk process is associated with or it is subject to one of the problems described in the preceding bullets.

Effect. The requested operation is not performed.

Recovery. Appropriate recovery varies according to the specific cause of the problem. Refer to the event log to determine appropriate action. The recovery might be, for example, to add physical volumes to the storage pool, bring volumes to the UP state, or remove the physical volume specification from the original request.

```
5015    (%11627) The outcome for the request is unknown.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. During processing of a file-system request (create, purge, rename, or alter) the NonStop Storage Management Foundation (SMF) subsystem lost communication with the disk process. This could occur, for example, as a result of a processor failure.

Effect. The requested operation is not performed.

Recovery. Take appropriate action to recover the disk process and then retry the request.

```
5017    (%11631) Unable to read one of the SMF catalog
                 tables.  (device type:  3.36, 25.0, or 52.0)
```

Cause. A process other than the NonStop Storage Management Foundation (SMF) subsystem attempted to directly access an SMF catalog file and a file-system error occurred.

Effect. The requested operation is not performed.

Recovery. Refer to the event log for detailed information and take appropriate action.

```
5018    (%11632) An error occurred during message processing.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. A file-system request (create, purge, or alter) was made against a NonStop Storage Management Foundation (SMF) object, and the reply from the disk process contained an error. This often indicates a HP internal error.

Effect. The requested operation is not performed.

Recovery. Refer to the event log for additional information and take action as appropriate. Retry the request. If the problem persists, contact your service provider.

```
5028    (%11644) Unable to read a received message.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. A NonStop Storage Management Foundation (SMF) process received an invalid message from any process.

Effect. The message is ignored and the requested operation is not performed.

Recovery. Refer to the event log for additional information and take action as appropriate. Retry the request. If the problem persists, contact your service provider.

```
5034    (%11652) Unable to start the thread manager.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. A request was made to a NonStop Storage Management Foundation (SMF) disk process or pool process, which was unable to start the thread manager while processing the request. This error is typically caused by a shortage of some resource, usually memory.

Effect. The requested operation is not performed.

Recovery. Refer to the event log for additional information and take action as appropriate. The recovery might be to wait and retry the request when the system is less busy. If the problem persists or if it occurs frequently, contact your service provider.

```
5035    (%11653) Unable to start a thread.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. A request was made to a NonStop Storage Management Foundation (SMF) disk process or pool process, which was unable to start a thread while processing the request. This error is typically caused by a shortage of some resource, usually memory.

Effect. The requested operation is not performed.

Recovery. Wait and retry the request when the system is less busy. If the problem persists or if it occurs frequently, contact your service provider.

```
5043    (%11663) An error occurred during a read of a SMF
                 catalog file.  (device type:  3.36, 25.0, or
                 52.0)
```

Cause. A file-system request (create, purge, rename, or alter) was made, and an error occurred during a read of a NonStop Storage Management Foundation (SMF) catalog file.

Effect. The requested operation is not performed.

Recovery. Appropriate recovery varies according to the specific file-system error that occurred during the read. Refer to the event log to determine the file-system error and take appropriate action.

```
5048    (%11670) BEGINTRANSACTION error.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. The NonStop Storage Management Foundation (SMF) subsystem was trying to begin a Transaction Management Facility (TMF) transaction during processing of the operation when an error occurred on the call to the BEGINTRANSACTION procedure.

Effect. The requested operation failed and recovery processing is aborted. The SMF subsystem periodically tries again to perform recovery processing.

Recovery. Appropriate recovery varies according to the specific error returned by the BEGINTRANSACTION procedure. Refer to the event log and take action as appropriate. The recovery might be to just retry the request. In some cases, no user action is required.

```
5049    (%11671) A memory allocation failed.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. The NonStop Storage Management Foundation (SMF) subsystem was unable to allocate memory due to a shortage of resources.

Effect. The requested operation is not performed.

Recovery. Wait and retry the request. If the problem persists, bring the SMF process to the DOWN state and then to the UP state before retrying the request again. If the problem still persists, contact your service provider.

```
5050    (%11672) An unexpected error occurred during request
                 processing.  (device type:  3.36, 25.0, or
                 52.0)
```

Cause. This error message indicates a HP internal error.

Effect. The requested operation is not performed.

Recovery. Refer to the event log for additional information and take action as appropriate. Retry the request. If the problem persists, contact your service provider.

```
5053    (%11675) The SMF lock manager reached its lock
                 threshold.  (device type:  3.36, 25.0, or
                 52.0)
```

Cause. The maximum number of locks have been granted on a particular NonStop Storage Management Foundation (SMF) object.

Effect. The requested operation is not performed.

Recovery. Wait and retry the request. If the problem persists, bring the SMF process to the DOWN state and then to the UP state before retrying the request again. If the problem still persists, contact your service provider.

```
5064    (%11710) An error occurred during recovery
                 processing.  (device type:  3.36, 25.0, or
                 52.0)
```

Cause. An error occurred while the NonStop Storage Management Foundation (SMF) subsystem was attempting to perform recovery processing.

Effect. The requested operation failed and recovery processing is aborted. The SMF subsystem periodically tries again to perform recovery processing.

Recovery. Appropriate recovery varies according to the specific error that occurred. Refer to the event log for detailed information. In some cases, no user action is required.

```
5065    (%11711) An error occurred during indeterminate
                 outcome processing.  (device type:  3.36,
                 25.0, or 52.0)
```

Cause. A resource allocation failure occurred within the NonStop Storage Management Foundation (SMF) subsystem during recovery processing after a 5015 error was returned.

Effect. The requested operation failed and recovery processing is aborted.

Recovery. Refer to the event log for additional information and take action as appropriate. Bring the SMF process to the DOWN state and then to the UP state. Retry the request. If the problem persists, contact your service provider.

```
5072    (%11720) Recovery processing was unable to initialize
                 PENDOPS processing.  (device type:  3.36,
                 25.0, or 52.0)
```

Cause. When a NonStop Storage Management Foundation (SMF) process attempted to perform recovery processing after the failure of a create, purge, rename, or update of an SMF object, an error occurred that prevented the SMF process from initializing the PENDOPS (pending operations) file. This error can also be returned if there is a lock failure during recovery processing or if the PENDOPS file is unavailable.

Effect. The requested operation failed and recovery processing is aborted. The SMF process enters the DOWN state.

Recovery. Refer to the event log for detailed information. In the case of a lock failure, take appropriate action. Otherwise, wait and retry the operation by bringing the SMF process to an UP state. If the problem persists, contact your service provider.

```
5073    (%11721) A PENDOPS semaphore failed; the process that
                 locked a SMF object could not be identified.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. When a NonStop Storage Management Foundation (SMF) process attempted to perform recovery processing after the failure of a create, purge, rename, or update of an SMF object, an error occurred in the PENDOPS (pending operations) file, which prevented identification of the process that locked an SMF object.

Effect. The requested operation failed and recovery processing is aborted. The SMF process enters the DOWN state.

Recovery. Wait and retry the operation by bringing the SMF process to an UP state. If the problem persists, contact your service provider.

```
5076    (%11724) An invalid operation code was found in the
                 PENDOPS table; the PENDOPS entry is invalid.
                 (device type:  3.36, 25.0, or 52.0)
```

Cause. When the NonStop Storage Management Foundation (SMF) subsystem attempted to perform recovery processing after an SMF process failed, an invalid operation code was found in the PENDOPS (pending operations) file.

Effect. The requested operation failed and recovery processing is aborted.

Recovery. Contact your service provider.

```
5313    (%12301) A virtual disk process encountered an error
                 when trying to perform the requested
                 operation.  (device type:  3.36)
```

Cause. During processing of a file-system request (create, open, purge, rename, alter, or information inquiry) on a NonStop Storage Management Foundation (SMF) logical file, the virtual disk process encountered an error.

Effect. The requested operation is not performed.

Recovery. Appropriate recovery varies according to the specific error. Refer the event log to determine the specific file-system error and take appropriate action.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a file-system error list in a response. HP subsystems return such an error list when, in performing your request, they call a file-system procedure directly or indirectly and an error occurs on the call.

The contents of the error list depend on which procedure was called. The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every file-system error list, identifies the procedure.

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list can include any of the conditional tokens listed under its description and in Table 2-4.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a file-system error list:

- Include all unconditional tokens listed in the error-list description.
- Make the ZFIL-TKN-XFILENAME token value null if you cannot obtain a valid file name, such as when these errors occur:
  - File-system error 16 (file not opened). This error indicates a coding error.
  - File-system error 26 (no outstanding I/O requests) when it is returned by an AWAITIO call in which *file-number* is equal to -1.
  - File-system error 40 (operation timed out) when it is returned by an AWAITIO call in which *file-number* is equal to -1.
- Optionally include any or none of the conditional tokens listed in Table 2-4. Note that the tokens listed might not be appropriate for every file-system error. To obtain values for the tokens, use FILE_GETINFOLIST_ or FILEINFO and pass the procedure a valid file number, or use FILE_GETINFOLISTBYNAME_ or FILEINFO and pass the procedure a valid file name.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, and for additional information about tokens and token types, and for definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

Table 2-4. Conditional Tokens That Can Appear in Any File-System Error List (page 1 of 3)

Token Name	Token Type	Description
ZFIL-TKN-ALTKEYPARAMS	ZSPI-TYP-BYTESTRING	is the alternate key parameters.
ZFIL-TKN-BLOCKLENGTH	ZSPI-TYP-UINT	is the data block length.
ZFIL-TKN-CURRENTKEYLENGTH	ZSPI-TYP-INT	is the current key length.

Table 2-4. Conditional Tokens That Can Appear in Any File-System Error List (page 2 of 3)

Token Name	Token Type	Description
ZFIL-TKN-CURRENTKEYSPEC	ZSPI-TYP-INT	is the current key specifier.
ZFIL-TKN-CURRENTKEYVALUE	ZSPI-TYP-BYTESTRING	is the current key value.
ZFIL-TKN-CURRENTPRIKEYLENGTH	ZSPI-TYP-INT	is the current primary key length.
ZFIL-TKN-CURRENTRECPOINTER	ZSPI-TYP-INT2	is the current record pointer (relative byte address).
ZFIL-TKN-DEVTYPE	ZSPI-TYP-INT	is the device type.
ZFIL-TKN-EOFPOINTER	ZSPI-TYP-INT2	is the end-of-file location (relative byte address).
ZFIL-TKN-ERRORDETAIL	ZFIL-TYP-INT	is the *error-detail* parameter where applicable.
ZFIL-TKN-ERRORPARTITION	ZSPI-TYP-INT	is the partition in error.
ZFIL-TKN-EXTENTSIZE	ZSPI-TYP-INT	is the file extent size.
ZFIL-TKN-FILE-OPEN-ACCESS	ZFIL-TYP-INT	is the *file-open-access* mode.
ZFIL-TKN-FILE-OPEN-EXCLUSION	ZFIL-TYP-INT	is the *file-open-exclusion* mode.
ZFIL-TKN-FILE-OPEN-NOWAIT	ZFIL-TYP-INT	is the *file-open-nowait* depth.
ZFIL-TKN-FILE-OPEN-OPTIONS	ZFIL-TYP-INT	is the *file-open-options* flags.
ZFIL-TKN-FILECODE	ZSPI-TYP-ENUM	is the file code.
ZFIL-TKN-FILENUMBER	ZSPI-TYP-INT	is the file number.
ZFIL-TKN-FILETYPE	ZSPI-TYP-ENUM	is the file type.
ZFIL-TKN-KEYINERROR	ZSPI-TYP-INT	is the key specifier that is in error.
ZFIL-TKN-KEYPARAMS	ZSPI-TYP-BYTESTRING	is the key parameters.
ZFIL-TKN-LASTMODTIME	ZSPI-TYP-TIMESTAMP	is the last modified timestamp.
ZFIL-TKN-LDEV	ZSPI-TYP-INT	is the logical device number.
ZFIL-TKN-MAXEXTENTS	ZSPI-TYP-INT	is the maximum number of extents.
ZFIL-TKN-MAXSIZE	ZSPI-TYP-INT2	is the maximum file size.
ZFIL-TKN-NEXTRECPOINTER	ZSPI-TYP-INT2	is the next record location (relative byte address).
ZFIL-TKN-NUMEXTENTS	ZSPI-TYP-INT	is the number of allocated extents.
ZFIL-TKN-NUMPARTITIONS	ZSPI-TYP-INT	is the number of partitions.
ZFIL-TKN-OBJECTFILE	ZSPI-TYP-FNAME	is the file name of the reporting program.

Table 2-4. Conditional Tokens That Can Appear in Any File-System Error List (page 3 of 3)

Token Name	Token Type	Description
ZFIL-TKN-OPENFLAGS	ZSPI-TYP-UINT	is the number of open flags.
ZFIL-TKN-OWNER	ZSPI-TYP-BYTE-PAIR	is the file owner ID or the CRAID (process ID).
ZFIL-TKN-PARTITIONPARAMS	ZSPI-TYP-BYTESTRING	is the partition parameters.
ZFIL-TKN-PARTITIONSIZE	ZSPI-TYP-INT	is the partition size.
ZFIL-TKN-RECORDLENGTH	ZSPI-TYP-INT	is the record length.
ZFIL-TKN-SECONDARYEXTENTSIZE	ZSPI-TYP-INT	is the secondary extent size.
ZFIL-TKN-SECURITY	ZSPI-TYP-ENUM	is the file security.
ZFIL-TKN-SUBDEVNUMBER	ZSPI-TYP-INT	is the subdevice number.
ZFIL-TKN-SYNCRECEIVEDEPTH	ZSPI-TYP-INT	is the sync or receive depth.
ZFIL-TKN-UNSTRUCTUREDBUFSIZE	ZSPI-TYP-INT	is the unstructured buffer size.

1: ZFIL-VAL-AWAITIO

A call to the file-system procedure AWAITIO resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR        token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED  token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME          token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG               token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TIMEOUT           token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-AWAITIO (1).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the I/O buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length in bytes of transferred data.

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TIMEOUT is the timeout value.

Effect

The AWAITIO operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

2: ZFIL-VAL-CHECKCLOSE

A call to the procedure FILE_CLOSE_CHKPT_ or CHECKCLOSE resulted in a condition code less (CCL). Either the file number supplied in the call is invalid, or the caller's backup no longer exists.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                  token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR                 token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR              token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME             token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST                  token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-FILENUMBER            token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME              token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAPEDISP              token-type ZSPI-TYP-ENUM.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always zero, since CHECKCLOSE does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKCLOSE (2).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-FILENUMBER is the number of the closed file.

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAPEDISP is the selected tape disposition.

Effect

The operation fails.

Recovery

If the file number is invalid, correct the file number. If the caller's backup no longer exists, this error is for information only; no corrective action is necessary.

3: ZFIL-VAL-CHECKMONITOR

The primary process paired with the process that called CHECKMONITOR stopped or executed a call to CHECKSWITCH.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-CHECKPOINTSTATUS token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-OLDPRIMARY       token-type ZSPI-TYP-CRTPID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always zero, since CHECKMONITOR does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKMONITOR (3).

ZFIL-TKN-CHECKPOINTSTATUS bits <8:15> indicate why the backup took over:

Value	Meaning
0	Primary stopped

Value	Meaning
1	Primary abnormally ended
2	Primary’s processor failed
3	Primary called CHECKSWITCH

Conditional Tokens

ZFIL-TKN-OLDPRIMARY is the old primary process ID.

Effect

Normal processing continues in the new primary.

Recovery

Informative message only; no corrective action is needed.

4: ZFIL-VAL-CHECKOPEN

A call to the procedure CHECKOPEN returned a nonzero file-system error code.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                 token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR                token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR             token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-FILENAME             token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST                 token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-FILENUMBER           token-type ZSPI-TYP-INT.
   ZFIL-TKN-OPENFLAGS            token-type ZSPI-TYP-UINT.
   ZFIL-TKN-SYNCRECEIVEDEPTH     token-type ZSPI-TYP-INT.
   ZFIL-TKN-BLOCKBUFFER          token-type ZSPI-TYP-INT.
   ZFIL-TKN-BLOCKBUFFERLENGTH    token-type ZSPI-TYP-UINT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKOPEN (4).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Conditional Tokens

ZFIL-TKN-FILENUMBER is the number of the file opened in the primary process.

ZFIL-TKN-OPENFLAGS is the open flags used by the primary process.

ZFIL-TKN-SYNCRECEIVEDEPTH is the sync or receive depth used by the primary process.

ZFIL-TKN-BLOCKBUFFER is the address of the sequential block buffer.

ZFIL-TKN-BLOCKBUFFERLENGTH is the length in bytes of the sequential block buffer.

Effect

The file might not be correctly opened in the primary or backup process of a process pair.

Recovery

Follow the recommended recovery procedure for the returned file-system error code.

5: ZFIL-VAL-CHECKPOINT

A call to the procedure CHECKPOINT returned a nonzero file-system error code. Either an error was detected, or a backup takeover occurred.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                 token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR                token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR             token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-CHECKPOINTSTATUS     token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST                 token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-STACKBASE            token-type ZSPI-TYP-UINT.
   ZFIL-TKN-SREGISTER            token-type ZSPI-TYP-UINT.
   ZFIL-MAP-CHECKPOINTCELL

      def ZFIL-DDL-CHECKPOINTCELL
         version C00
         FOR z-file-sync THROUGH z-file-number
      end.

   ZFIL-TKN-OLDPRIMARY           token-type ZSPI-TYP-CRTPID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always zero, since CHECKPOINT does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKPOINT (5).

ZFIL-TKN-CHECKPOINTSTATUS indicates the status of the call to CHECKPOINT. This token returns a status word in the following form:

Value	Meaning
<0:7> = 0	No error
<0:7> = 1	There is no backup or the primary process cannot communicate with the backup; then <8:15> = file-system error number

<0:7> = 2	Takeover from the primary; then:
	  <8:15> = 0 (primary stopped)
	  <8:15> = 1 (primary abnormally ended)
	  <8:15> = 2 (primary's processor failed)
	  <8:15> = 3 (primary called CHECKSWTICH)

Conditional Tokens

ZFIL-TKN-STACKBASE is the checkpoint base-of-stack.

ZFIL-TKN-SREGISTER is the current S register.

ZFIL-MAP-CHECKPOINTCELL is the data to checkpoint. This token can appear from 1 to 13 times.

ZFIL-TKN-OLDPRIMARY is the old primary process ID.

Effect

If an error occurred, the operation fails.

If the backup process took over, the system continues normally.

Recovery

Check ZFIL-TKN-CHECKPOINTSTATUS for information about CHECKPOINT.

If the backup process took over, this is an informative message only; no corrective action is necessary.

6: ZFIL-VAL-CHECKPOINTMANY

A call to the procedure CHECKPOINTMANY returned a nonzero file-system error code. Either an error was detected or a backup takeover occurred.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                 token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR                token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR             token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-CHECKPOINTSTATUS     token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST                 token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-STACKBASE            token-type ZSPI-TYP-UINT.
   ZFIL-TKN-SREGISTER            token-type ZSPI-TYP-UINT.
   ZFIL-MAP-CHECKPOINTLIST

      def ZFIL-DDL-CHECKPOINTLIST
         version C00
         FOR z-cellcount THROUGH z-checkpointcell
      end.

   ZFIL-TKN-OLDPRIMARY           token-type ZSPI-TYP-CRTPID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always zero, since CHECKPOINTMANY does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKPOINTMANY (6).

ZFIL-TKN-CHECKPOINTSTATUS indicates the status of the call to CHECKPOINTMANY. This token returns a status word in the following form:

Value	Meaning
<0:7> = 0	No error
<0:7> = 1	There is no backup or the primary process cannot communicate with the backup; then <8:15> = file-system error number
<0:7> = 2	Takeover from the primary; then <8:15> = 0 (primary stopped) <8:15> = 1 (primary abnormally failed) <8:15> = 2 (primary's processor failed) <8:15> = 3 (primary called CHECKSWITCH)
<0:7> = 3	Invalid parameter; then <8:15> = 1 (error in *stack-base* parameter) <8:15> = n, n > 1 (error in *word*[*n*-2]; refer to the *Guardian Procedure Calls Reference Manual* for more information

Conditional Tokens

ZFIL-TKN-STACKBASE is the checkpoint base-of-stack.

ZFIL-TKN-SREGISTER is the current S register.

ZFIL-MAP-CHECKPOINTLIST is the CHECKPOINT list passed to CHECKPOINTMANY.

ZFIL-TKN-OLDPRIMARY is the old primary process ID.

Effect

The operation fails. If the backup process took over, the system continues normally.

Recovery

If an error occurred, check ZFIL-VAL-CHECKPOINTSTATUS for information about CHECKPOINTMANY.

If the backup process took over, this is an informative message only; no corrective action is necessary.

7: ZFIL-VAL-CHECKSWITCH

A call to the procedure CHECKSWITCH returned a nonzero file-system error code. Either an error was detected or a backup takeover occurred.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-CHECKPOINTSTATUS    token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-OLDPRIMARY          token-type ZSPI-TYP-CRTPID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always is zero, since CHECKPOINT does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKSWITCH (7).

ZFIL-TKN-CHECKPOINTSTATUS indicates the status of the call to CHECKSWITCH. This token returns a status word in the following form:

Value	Meaning
<0:7> = 0	No error
<0:7> = 1	There is no backup or the primary process cannot communicate with the backup; then <8:15> = file-system error number
<0:7> = 2	Takeover from the primary; then <8:15> = 0 primary stopped <8:15> = 1 primary abnormally ended <8:15> = 2 primary's processor failed <8:15> = 3 primary called CHECKSWITCH

Conditional Tokens

ZFIL-TKN-OLDPRIMARY is the old primary process ID.

Effect

If an error occurred, the operation fails. If the backup process took over, the system continues normally.

Recovery

Check ZFIL-VAL-CHECKPOINTSTATUS for information about CHECKSWITCH. If the backup process took over, this is an informative message only; no corrective action is necessary.

8: ZFIL-VAL-CONTROL

A call to the procedure CONTROL resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR        token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR     token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME    token-type ZSPI-TYP-STRING.
   ZFIL-TKN-OPERATION    token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-FILENAME     token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-PARAMETER    token-type ZSPI-TYP-INT.
   ZFIL-TKN-TAG          token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CONTROL (8).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

ZFIL-TKN-OPERATION is the CONTROL operation code. Refer to the *Guardian Procedure Calls Reference Manual* for information about CONTROL operation codes.

Conditional Tokens

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-PARAMETER is the CONTROL operation parameter. Refer to the *Guardian Procedure Calls Reference Manual* for information about the CONTROL operation parameter.

ZFIL-TKN-TAG is the AWAITIO tag value.

Effect

The attempted CONTROL operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

9: ZFIL-VAL-CREATE

A call to the procedure CREATE resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-FILENAME        token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-OPENDEFAULTS    token-type ZSPI-TYP-UINT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CREATE (9).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Conditional Tokens

ZFIL-TKN-OPENDEFAULTS is the file-label open defaults.

Effect

The attempted CREATE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

10: ZFIL-VAL-KEYPOSITION

A call to the procedure KEYPOSITION resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
 ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
 ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
 ZFIL-TKN-XFILENAME          token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-COMPAREKEYLENGTH token-type ZSPI-TYP-BYTE-PAIR.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-KEYVALUE         token-type ZSPI-TYP-BYTESTRING.
   ZFIL-TKN-KEYSPEC          token-type ZSPI-TYP-INT.
   ZFIL-TKN-POSITIONMODE     token-type ZSPI-TYP-ENUM.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-KEYPOSITION (10).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-COMPAREKEYLENGTH is the key length (<8:15>) and the compare length (<0:7>).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-KEYVALUE is the key value.

ZFIL-TKN-KEYSPEC is the key specifier.

ZFIL-TKN-POSITIONMODE is the positioning mode.

Effect

The attempted KEYPOSITION operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

11: ZFIL-VAL-OPEN

A call to the OPEN procedure resulted in a condition code less (CCL) or a condition code greater (CCG). If CCL (ZFIL-TKN-CONDITION = -1), the file is not opened. If CCG (ZFIL-TKN-CONDITION = 1), the file is open, but the file system detected an exceptional condition during the OPEN operation.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR                token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PROC-ERR             token-type ZSPI-TYP-ENUM.
  ZFIL-TKN-FILENAME             token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
  ZFIL-TKN-CONDITION            token-type ZSPI-TYP-INT.
  ZFIL-TKN-PRIMARYFILE          token-type ZSPI-TYP-INT.
  ZFIL-TKN-PRIMARYPROCESS       token-type ZSPI-TYP-CRTPID.
  ZFIL-TKN-BLOCKBUFFERLENGTH    token-type ZSPI-TYP-UINT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-OPEN (11).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Conditional Tokens

ZFIL-TKN-CONDITION is the condition code returned by OPEN (-1, 0, or 1).

ZFIL-TKN-PRIMARYFILE is the primary file number.

ZFIL-TKN-PRIMARYPROCESS is the primary process ID of the process for which the backup process reports the error.

ZFIL-TKN-BLOCKBUFFERLENGTH is the length of the requested sequential block buffer.

Effect

The attempted OPEN operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

12: ZFIL-VAL-PURGE

A call to the PURGE procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-FILENAME      token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-PURGE (12).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Effect

The attempted PURGE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

13: ZFIL-VAL-POSITION

A call to the POSITION procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME     token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-FILENAME      token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-RECORDSPEC    token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-POSITION (13).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-RECORDSPEC is the record specifier (RBA).

Effect

The attempted POSITION operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

14: ZFIL-VAL-READ

A call to the READ procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST               token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME          token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST               token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR         token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED   token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME           token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT      token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READ (14).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the READ buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted read operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

15: ZFIL-VAL-READLOCK

A call to the READLOCK procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST               token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME          token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST               token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR         token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED   token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME           token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT      token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READLOCK (15).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the READLOCK buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READLOCK operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

16: ZFIL-VAL-READUPDATE

A call to the READUPDATE procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR       token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG              token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT    token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READUPDATE (16).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the READUPDATE buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READUPDATE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

17: ZFIL-VAL-READUPDATELOCK

A call to the READUPDATELOCK procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR       token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG              token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT    token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READUPDATELOCK (17).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the READUPDATELOCK buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READUPDATELOCK operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

18: ZFIL-VAL-REPLY

A call to the REPLY procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-FILENAME          token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR        token-type ZSPI-TYP-UINT.
   ZFIL-TKN-TRANSFERCOUNT     token-type ZSPI-TYP-INT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-RECEIVETAG        token-type ZSPI-TYP-INT2.
   ZFIL-TKN-REPLYCODE         token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-REPLY (18).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the REPLY buffer.

ZFIL-TKN-TRANSFERCOUNT is the length of REPLY (in bytes).

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-RECEIVETAG is the LASTRECEIVE or RECEIVEINFO tag value.

ZFIL-TKN-REPLYCODE is the error value returned with the reply.

Effect

The attempted REPLY operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

19: ZFIL-VAL-SETMODE

A call to the SETMODE procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME     token-type ZSPI-TYP-STRING.
   ZFIL-TKN-FUNCTION      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-FILENAME      token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-PARAM1        token-type ZSPI-TYP-INT.
   ZFIL-TKN-PARAM2        token-type ZSPI-TYP-INT.
   ZFIL-TKN-LASTPARAMS    token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-SETMODE (19).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

ZFIL-TKN-FUNCTION is the SETMODE function. Refer to the *Guardian Procedure Calls Reference Manual* for more information about the SETMODE functions.

Conditional Tokens

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-PARAM1 is the first function parameter.

ZFIL-TKN-PARAM2 is the second function parameter.

ZFIL-TKN-LASTPARAMS contains the previous parameter settings.

Effect

The attempted SETMODE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

20: ZFIL-VAL-SETMODENOWAIT

A call to the SETMODENOWAIT procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME     token-type ZSPI-TYP-STRING.
   ZFIL-TKN-FUNCTION      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-FILENAME      token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-PARAM1        token-type ZSPI-TYP-INT.
   ZFIL-TKN-PARAM2        token-type ZSPI-TYP-INT.
   ZFIL-TKN-LASTPARAMS    token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TAG           token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-SETMODENOWAIT (20).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

ZFIL-TKN-FUNCTION is the SETMODENOWAIT function. Refer to the *Guardian Procedure Calls Reference Manual* for more information about the SETMODENOWAIT functions.

Conditional Tokens

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-PARAM1 is the first function parameter.

ZFIL-TKN-PARAM2 is the second function parameter.

ZFIL-TKN-LASTPARAMS contains the previous parameter settings.

ZFIL-TKN-TAG is the AWAITIO tag value.

Effect

The attempted SETMODENOWAIT operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

21: ZFIL-VAL-WRITE

A call to the WRITE procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR        token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME          token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG               token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT     token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITE (21).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the WRITE buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted WRITE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

22: ZFIL-VAL-WRITEREAD

A call to the WRITEREAD procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR       token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-READCOUNT        token-type ZSPI-TYP-INT.
   ZFIL-TKN-TAG              token-type ZSPI-TYP-INT2.
   ZFIL-TKN-WRITECOUNT       token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEREAD (22).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the WRITEREAD buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-READCOUNT is the read length (in bytes).

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-WRITECOUNT is the write length (in bytes).

Effect

The attempted WRITEREAD operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

23: ZFIL-VAL-WRITEUPDATE

A call to the WRITEUPDATE procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR        token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME          token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG               token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT     token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEUPDATE (23).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the WRITEUPDATE buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted WRITEUPDATE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

24: ZFIL-VAL-WRITEUPDATEUNLOCK

A call to the WRITEUPDATEUNLOCK procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME           token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDR          token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED    token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME            token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                 token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT       token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEUPDATEUNLOCK (24).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDR is the base address of the WRITEUPDATEUNLOCK buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted WRITEUPDATEUNLOCK operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

25: ZFIL-VAL-AWAITIOX

A call to the file-system procedure AWAITIOX resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME           token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX         token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED    token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME            token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                 token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TIMEOUT             token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-AWAITIOX (25).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the I/O buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the transferred data (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIOX tag value.

ZFIL-TKN-TIMEOUT is the timeout value.

Effect

The AWAITIOX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

26: ZFIL-VAL-CHECKPOINTMANYX

A call to the procedure CHECKPOINTMANYX returned a nonzero file-system error. Either an error was detected or a backup takeover occurred.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-CHECKPOINTSTATUS    token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-STACKBASE           token-type ZSPI-TYP-UINT.
   ZFIL-TKN-SREGISTER           token-type ZSPI-TYP-UINT.
   ZFIL-MAP-CHECKPOINTLIST

      def ZFIL-DDL-CHECKPOINTLIST
         version C00
         FOR z-cellcount THROUGH z-checkpointcell
      end.

   ZFIL-TKN-OLDPRIMARY          token-type ZSPI-TYP-CRTPID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always zero, since CHECKPOINTMANYX does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKPOINTMANYX (26). *ZFIL-TKN-CHECKPOINTSTATUS* indicates the status of the call to CHECKPOINTMANYX. This token returns a status word in the following form:

Value	Meaning
<0:7> = 0	No error
<0:7> = 1	There is no backup or the primary process cannot communicate with the backup; then <8:15> = file-system error number
<0:7> = 2	Takeover from the primary; then <8:15> = 0 primary stopped <8:15> = 1 primary abnormally ended <8:15> = 2 primary's processor failed <8:15> = 3 primary called CHECKSWITCH
<0:7> = 3	Invalid parameter; then <8:15> = 1 error in *stack-base* parameter <8:15> = n, n > 1 error in *word*[*n*-2]; refer to the *Guardian Procedure Calls Reference Manual* for more information

Conditional Tokens

ZFIL-TKN-STACKBASE is the checkpoint base-of-stack.

ZFIL-TKN-SREGISTER is the current S register.

ZFIL-MAP-CHECKPOINTLIST is the CHECKPOINTX list passed to CHECKPOINTMANYX.

ZFIL-TKN-OLDPRIMARY is the old primary process ID.

Effect

If an error occurs, the operation fails. If the backup process takes over, the system continues normally.

Recovery

Check ZFIL-TKN-CHECKPOINTSTATUS for information about CHECKPOINTMANYX.

If the backup process took over, this message is informative only; no corrective action is necessary.

27: ZFIL-VAL-CHECKPOINTX

A call to the procedure CHECKPOINTX returned a nonzero file-system error. Either an error was detected or a backup takeover occurred.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-CHECKPOINTSTATUS    token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-STACKBASE           token-type ZSPI-TYP-UINT.
   ZFIL-TKN-SREGISTER           token-type ZSPI-TYP-UINT.
   ZFIL-MAP-CHECKPOINTCELL

      def ZFIL-DDL-CHECKPOINTCELL
         version C00
         FOR z-file-sync THROUGH z-file-number
      end.

   ZFIL-TKN-OLDPRIMARY          token-type ZSPI-TYP-CRTPID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO; it is always zero, since CHECKPOINTX does not return an error code.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CHECKPOINTX (27).

ZFIL-TKN-CHECKPOINTSTATUS indicates the status of the call to CHECKPOINTX. This token returns a status word in the following form:

Value	**Meaning**
<0:7> = 0	No error
<0:7> = 1	There is no backup or the primary process cannot communicate with the backup; then <8:15> = file-system error number
<0:7> = 2	Takeover from the primary; then <8:15> = 0 primary stopped <8:15> = 1 primary abnormally ended <8:15> = 2 primary’s processor failed <8:15> = 3 primary called CHECKSWITCH

Conditional Tokens

ZFIL-TKN-STACKBASE is the checkpoint base-of-stack.

ZFIL-TKN-SREGISTER is the current S register.

ZFIL-MAP-CHECKPOINTCELL is the data to checkpoint. This token can appear from 1 to 13 times.

ZFIL-TKN-OLDPRIMARY is the old primary process ID.

Effect

If an error occurs, the operation fails. If the backup process takes over, the system continues normally.

Recovery

Check ZFIL-TKN-CHECKPOINTSTATUS for information about CHECKPOINTX.

If the backup process took over, this message is informative only; no corrective action is necessary.

28: ZFIL-VAL-KEYPOSITIONX

A call to the procedure KEYPOSITIONX resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-COMPAREKEYLENGTH token-type ZSPI-TYP-BYTE-PAIR.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-KEYVALUE         token-type ZSPI-TYP-BYTESTRING.
   ZFIL-TKN-KEYSPEC          token-type ZSPI-TYP-INT.
   ZFIL-TKN-POSITIONMODE     token-type ZSPI-TYP-ENUM.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-KEYPOSITIONX (28).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-COMPAREKEYLENGTH is the key length (<8:15>) and the compare length (<0:7>).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-KEYVALUE is the key value.

ZFIL-TKN-KEYSPEC is the key specifier.

ZFIL-TKN-POSITIONMODE is the positioning mode.

Effect

The attempted KEYPOSITIONX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

29: ZFIL-VAL-READLOCKX

A call to the READLOCKX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX      token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG              token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT    token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READLOCKX (29).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the READLOCKX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READLOCKX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

30: ZFIL-VAL-READUPDATELOCKX

A call to the READUPDATELOCKX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX      token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG              token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT    token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READUPDATELOCKX (30).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the READUPDATELOCKX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READUPDATELOCKX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

31: ZFIL-VAL-READUPDATEX

A call to the READUPDATEX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX       token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME          token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG               token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT     token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READUPDATEX (31).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the READUPDATEX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READUPDATEX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

32: ZFIL-VAL-READX

A call to the READX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME           token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX         token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED    token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME            token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                 token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT       token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-READX (32).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the READX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted READX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

33: ZFIL-VAL-REPLYX

A call to the REPLYX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX      token-type ZSPI-TYP-UINT.
   ZFIL-TKN-TRANSFERCOUNT    token-type ZSPI-TYP-INT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-RECEIVETAG       token-type ZSPI-TYP-INT2.
   ZFIL-TKN-REPLYCODE        token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-REPLYX (33).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the REPLYX buffer.

ZFIL-TKN-TRANSFERCOUNT is the length of the reply (in bytes).

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-RECEIVETAG is the LASTRECEIVE or RECEIVEINFO tag value.

ZFIL-TKN-REPLYCODE is the error value returned with the reply.

Effect

The attempted REPLYX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

34: ZFIL-VAL-WRITEREADX

A call to the WRITEREADX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX       token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED  token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME          token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-READCOUNT         token-type ZSPI-TYP-INT.
   ZFIL-TKN-TAG               token-type ZSPI-TYP-INT2.
   ZFIL-TKN-WRITECOUNT        token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEREADX (34).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the WRITEREADX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data read (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-READCOUNT is the read length (in bytes).

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-WRITECOUNT is the write length (in bytes).

Effect

The attempted WRITEREADX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

35: ZFIL-VAL-WRITEUPDATEUNLOCKX

A call to the WRITEUPDATEUNLOCKX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST               token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME          token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST               token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX        token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED   token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME           token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT      token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEUPDATEUNLOCKX (35).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the WRITEUPDATEUNLOCKX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted WRITEUPDATEUNLOCKX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

36: ZFIL-VAL-WRITEUPDATEX

A call to the WRITEUPDATEX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX      token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME         token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG              token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT    token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEUPDATEX (36).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the WRITEUPDATEX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted WRITEUPDATEX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

37: ZFIL-VAL-WRITEX

A call to the WRITEX procedure resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST               token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME          token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST               token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZFIL-TKN-BUFFERADDRX        token-type ZSPI-TYP-UINT.
   ZFIL-TKN-COUNTTRANSFERRED   token-type ZSPI-TYP-INT.
   ZFIL-TKN-FILENAME           token-type ZSPI-TYP-FNAME.
   ZFIL-TKN-TAG                token-type ZSPI-TYP-INT2.
   ZFIL-TKN-TRANSFERCOUNT      token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-WRITEX (37).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-BUFFERADDRX is the base address of the WRITEX buffer.

ZFIL-TKN-COUNTTRANSFERRED is the length of the data written (in bytes).

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

ZFIL-TKN-TAG is the AWAITIO tag value.

ZFIL-TKN-TRANSFERCOUNT is the request length (in bytes).

Effect

The attempted WRITEX operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

38: ZFIL-VAL-CLOSE

A call to the file-system procedure FILE_CLOSE_ or CLOSE resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
ZSPI-TKN-FILENAME             token-type ZSPI-TYP-FNAME.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-CLOSE (38).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Conditional Tokens

ZFIL-TKN-FILENAME is the name of the file in internal network format. If the file name cannot be represented in this format or if the file name is not known (file-system error 16 occurred), this token will not appear.

Effect

The CLOSE operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

39: ZFIL-VAL-DEVICEINFO2

A call to the file-system procedure FILE_GETINFOBYNAME_ or DEVICEINFO2 resulted in an unexpected error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                 token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR                token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR             token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-FILENAME             token-type ZSPI-TYP-FNAME.
ZSPI-TKN-ENDLIST                 token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the file-system error code returned in the *error* parameter of FILEINFO.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-DEVICEINFO2 (39).

ZFIL-TKN-FILENAME is the file name (blank-filled if unknown).

Effect

The DEVICEINFO2 operation fails.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

65: ZFIL-VAL-FILE-OPEN-CHKPT

A call to FILE_OPEN_CHKPT_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
   ZFIL-TKN-STATUS         token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_OPEN_CHKPT_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-OPEN-CHKPT (65).

ZFIL-TKN-XFILENAME is the file name.

ZFIL-TKN-STATUS qualifies the source of the error.

Effect

Status is the reason for the non zero value of the *error* parameter of FILE_OPEN_CHKPT_ and is the code returned in the *status* parameter to the procedure. FILE_OPEN_CHKPT_ failed for the reason denoted by a non zero status value with an error value as in the previously described error codes.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

66: ZFIL-VAL-FILE-CREATELIST

A call to FILE_CREATELIST_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_CREATELIST_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-CREATELIST (66).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

67: ZFIL-VAL-FILE-OPEN

A call to FILE_OPEN_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME         token-type ZSPI-TYP-STRING.
   ZFIL-TKN-STATUS            token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_OPEN_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-OPEN (67).

ZFIL-TKN-XFILENAME is the file name.

ZFIL-TKN-STATUS qualifies the source of the error.

Effect

Status is the file number returned by FILE_OPEN_. If an error occurred and ZFIL-TKN-STATUS is -1, the open failed. If ZFIL-TKN-STATUS is not -1, the file was opened with a warning.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

68: ZFIL-VAL-FILE-PURGE

A call to FILE_PURGE_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME     token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_PURGE_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-PURGE (68).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

69: ZFIL-VAL-FILE-CLOSE

A call to FILE_CLOSE_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME     token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_CLOSE_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-CLOSE (69).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

70: ZFIL-VAL-FILE-GETINFOBYNAME

A call to FILE_GETINFOBYNAME_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME     token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_GETINFOBYNAME_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-GETINFOBYNAME (70).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

71: ZFIL-VAL-FILE-GETRECEIVEINFO

A call to FILE_GETRECEIVEINFO_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_GETRECEIVEINFO_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-GETRECEIVEINFO (71).

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

72: ZFIL-VAL-FILENAME-COMPARE

A call to FILENAME_COMPARE_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILENAME_COMPARE_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILENAME-COMPARE (72).

ZFIL-TKN-XFILENAME (first occurrence) is the name of the first file.

ZFIL-TKN-XFILENAME (second occurrence) is the name of the second file.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

73: ZFIL-VAL-FILE-GETOPENINFO

A call to FILE_GETOPENINFO_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_GETOPENINFO_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-GETOPENINFO (73).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

74: ZFIL-VAL-DISK-REFRESH

A call to DISK_REFRESH_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME       token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of DISK_REFRESH_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-DISK-REFRESH (74).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

75: ZFIL-VAL-FILE-RENAME

A call to FILE_RENAME_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME        token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
ZFIL-TKN-XFILENAME           token-type ZSPI-TYP-STRING.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_RENAME_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-RENAME (75).

ZFIL-TKN-XFILENAME is the new file name.

Conditional Tokens

ZFIL-TKN-XFILENAME is the original file name.

Effect

The file is not renamed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

76: ZFIL-VAL-FILENAME-FINDSTART

A call to FILENAME_FINDSTART_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILENAME_FINDSTART_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILENAME-FINDSTART (76).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

77: ZFIL-VAL-FILENAME-FINDNEXT

A call to FILENAME_FINDNEXT_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILENAME_FINDNEXT_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILENAME-FINDNEXT (77).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format. The token ZFIL-TKN-XFILENAME can be null if certain errors occur. For example, if file-system error 16 (file not opened) occurs, the token will be null.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

78: ZFIL-VAL-FILENAME-FINDFINISH

A call to FILENAME_FINDFINISH_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR         token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILENAME_FINDFINISH_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILENAME-FINDFINISH (78).

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

80: ZFIL-VAL-FILE-CREATE

A call to FILE_CREATE_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
   ZFIL-TKN-XFILENAME       token-type ZSPI-TYP-STRING.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZFIL-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of FILE_CREATE_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZFIL-VAL-FILE-CREATE (80).

ZFIL-TKN-XFILENAME is the completely qualified file name (including node name) in external format.

Effect

The operation failed.

Recovery

Follow the recovery procedure for the returned file-system error code as described earlier in this section.

3 Sequential I/O Errors

The following error codes are produced by the sequential I/O (SIO) procedures. An application process can use the following SIO procedures to sequentially access files: CHECK^BREAK, CHECK^FILE, CLOSE^FILE, GIVE^BREAK, NO^ERROR, OPEN^FILE, READ^FILE, SET^FILE, TAKE^BREAK, WAIT^FILE, and WRITE^FILE. The error number is returned as the function result from the procedure call. Except for errors 521, 532, and 533, all the error codes in this section indicate fatal coding errors.

Interactive users can obtain a short explanation of most SIO errors by entering the following from the TACL prompt:

```
1> ERROR number
```

For further information about the sequential I/O (SIO) procedures, refer to the *Guardian Programmer's Guide* and the *Guardian Procedure Calls Reference Manual*.

Message Descriptions

This subsection lists each SIO error code and provides a description of each code.

```
512     SIOERR^INVALIDPARAM: An invalid parameter was
        specified.
```

Cause. The SIO procedure call contained an invalid parameter.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the parameter in error.

```
513     SIOERR^MISSINGFILENAME: A file name was missing.
```

Cause. The file control block (FCB) did not contain a file name.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. To correct this error, assign a file name before calling OPEN^FILE. You can assign a file name by using the ALLOCATE^FCB macro, the ALLOCATE^FCB^D00 macro, the SET^FILE procedure, or the command interpreter ASSIGN command.

```
514     SIOERR^DEVNOTSUPPORTED: The SIO procedures do not
        support the specified device type.
```

Cause. SIO procedures do not support the specified device type. This error is returned only by OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Do not attempt an SIO operation on devices of this type.

```
515     SIOERR^INVALIDACCESS: The specified access is
invalid.
```

Cause. This error results if:

- Meaningless access was specified (for example, a read from $0 or a line printer)
- Read-write access was specified for an EDIT file or in conjunction with blocking.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Do not attempt the SIO operations listed above.

```
516     SIOERR^INVALIDBUFADDR: The specified buffer address
is
        invalid.  A valid address is within 'G' [0:32767].
```

Cause. The buffer was not within 'G'[0:32767] of the user data area. This error is returned by OPEN^FILE if the block buffer fails the test, or by READ^FILE and WRITE^FILE if the data buffer fails the test.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Move the buffer into the lower half of the user data area.

```
517     SIOERR^INVALIDFILECODE: The specified file code in
the
        SET^FILE call does not match the file code of the
        file.
```

Cause. The file code specified in the ASSIGN command or in the ASSIGN^FILECODE option of the SET^FILE call did not match the file code of the file. This error is returned only by OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Change the file name or the file code in the ASSIGN command or in the ASSIGN^FILECODE option of the SET^FILE call.

```
518     SIOERR^BUFTOOSMALL: The block buffer provided to
        OPEN^FILE is too small.
```

Cause. The specified buffer was too small. This error is returned only by OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. For reading an EDIT file, allocate at least 144 bytes of buffer space.

For writing an EDIT file, allocate at least 1024 bytes of buffer space.

For blocking, allocate at least the same number of bytes for buffer space as the logical record length.

```
519     SIOERR^INVALIDBLKLENGTH: The block length specified
        in the SET^FILE call does not match the block buffer
        length in OPEN^FILE.
```

Cause. The ASSIGN block length or the ASSIGN^BLOCKLENGTH option of the SET^FILE call was greater than the block buffer length in the OPEN^FILE call. This error is only returned by OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the ASSIGN command or the ASSIGN^BLOCKLENGTH option, or use a larger buffer.

```
520     SIOERR^INVALIDRECLENGTH: The specified record length
        was either too small or too large.
```

Cause. There are three possible causes:

- The specified record length was zero or greater than the *max-record-length* specified in the OPEN^FILE call.
- The record length for the $RECEIVE file was less than 14.
- The record length was greater than 254 and the procedure specifies variable-length records.

This error is returned only by OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the record length.

```
521     SIOERR^INVALIDEDITFILE: The specified file is not
        a valid EDIT file.
```

Cause. The directory indicates that the EDIT file is damaged. This error is returned only by OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Refer to Section 15, EDITREAD and EDITREADINIT Errors, for corrective action.

```
522     SIOERR^FILEALREADYOPEN: Either the SET^FILE or
        CHECK^FILE operation is not valid on an open file
        or OPEN^FILE was called for a file already open.
```

Cause. The program used SET^FILE or CHECK^FILE for a file that should be closed, or it used OPEN^FILE for a file that is already open. This error is returned only by SET^FILE, CHECK^FILE, and OPEN^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Either close the file or correct the procedure call (for example, change parameters to permit operation when the file is open).

```
523     SIOERR^EDITREADERR: An EDITREAD or EDITREADINIT
        error occurred.
```

Cause. An EDITREAD or EDITREADINIT error occurred. This error is returned only by OPEN^FILE and READ^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Refer to Section 15, EDITREAD and EDITREADINIT Errors, for corrective action.

```
SIOERR^FILENOTOPEN: The specified file was not open.
```

Cause. The specified file is closed. There is a check, read, set, write, or wait error.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Either open the file or correct the procedure call (for example, change parameters to permit operation when the file is closed).

```
525     SIOERR^ACCESSVIOLATION: The requested operation was
        inconsistent with the access mode.
```

Cause. The operation was inconsistent with the access mode specified at OPEN^FILE time. For example, WRITE^FILE to a file opened for READACCESS only, or READ^FILE from a file opened for WRITEACCESS only.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Change the operation or access mode.

```
526     SIOERR^NOSTACKSPACE: The required operation failed
        because of insufficient stack space.
```

Cause. A waited WRITE^FILE operation attempted to allocate a temporary buffer from which to do the actual WRITE. For example, it needed to add padding beyond the end of the user-supplied buffer, or the next physical transfer began at an odd byte. However, the attempt failed. This error is returned only by WRITE^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Increase the run-time memory size if it is less than 32K bytes; otherwise, move one or more nonstring arrays to upper memory.

```
527     SIOERR^BLOCKINGREQD: The temporary buffer required
        for a nowait WRITE^FILE operation was not provided.
```

Cause. A nowait WRITE^FILE operation required a temporary buffer but none was provided.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Supply a block buffer.

```
530     SIOERR^INVALIDRECVWRITE: The program called
WRITE^FILE
        for $RECEIVE before calling READ^FILE.
```

Cause. The program called WRITE^FILE for $RECEIVE without first calling READ^FILE. This error is returned only by WRITE^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Add the missing read.

```
531     SIOERR^CANTOPENRECV: A call to CHECK^BREAK could not
        open $RECEIVE or $RECEIVE was opened without calling
        OPEN^FILE.
```

Cause. A call to CHECK^BREAK could not open $RECEIVE for break monitoring. The user opened $RECEIVE without calling the OPEN^FILE procedure. This error is returned only by CHECK^BREAK.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Change the program so that a call to OPEN^FILE opens $RECEIVE.

```
532     SIOERR^IORESTARTED: A nowait I/O operation has
        been restarted.
```

Cause. A nowait I/O was specified, and the first call to READ^FILE or WRITE^FILE started the first physical I/O. The first call to WAIT^FILE waits for the first physical operation to finish, starts the second, and returns this error. Subsequent calls to WAIT^FILE repeat this pattern until the logical I/O operation is finished and either or an actual error code is returned. READ^FILE, WRITE^FILE, and WAIT^FILE return this error.

Effect. The nowait I/O operation is restarted.

Recovery. Call WAIT^FILE again to continue waiting.

```
533     SIOERR^INTERNAL: An internal SIO error occurred.
```

Cause. An internal error occurred. This error is returned by WAIT^FILE and CLOSE^FILE.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Contact your service provider.

```
534     SIOERR^CHECKSUMCOMM: A discrepancy was detected
        between the common FCB checksum and the previous
        checksum.
```

Cause. While performing a checksum on the common file control block (FCB), the system encountered a discrepancy between this checksum and the previous checksum. This error is returned by all procedures.

Effect. The procedure sets the error code and returns without performing the requested operation and the process abends. The FCB might be damaged.

Recovery. Check the program for pointer errors or other errors that might have caused this error.

```
535     SIOERR^CHECKSUM: A discrepancy was detected
        between the file FCB checksum and the previous
        checksum.
```

Cause. While performing a checksum on the file control block (FCB), the system encountered a discrepancy between this checksum and the previous checksum. This error is returned by all procedures.

Effect. The procedure sets the error code and returns without performing the requested operation and the process abends. The FCB might be damaged.

Recovery. Check the program for pointer errors or other errors that could have returned this error.

```
536     SIO^OLDCOMMFCB: The FCB format allocated is not
         valid.
```

Cause. The file FCB used to open $RECEIVE is in the new D-series format, but the common FCB is in the old C-series or earlier format.

Effect. If the open flag, ABORT^OPENERR is set or defaults to TRUE, then the process ABENDs; if ABORT^OPENERR is set to FALSE, then error 536 is returned to the caller of OPEN^FILE. This is a fatal error.

Recovery. This error probably results from a programming error. Define the common FCB using the compile-time DEFINE^CBS^D or the run-time SET^FILE INIT^FILEFCB^D.

```
537     SIOERR^EDITLINEOFLOW: The highest possible EDIT line
        number was exceeded.
```

Cause. The SIO procedure call attempted to write a line to an EDIT file with a line number greater than 99999.999 (the largest allowable line number).

Effect. The line is not written. The error is treated as fatal.

Recovery. Write smaller files, use a file organization other than EDIT files, use SIO or IOEdit to specify fractional line numbers, or use IOEdit to renumber the file. The smallest possible line number increment that SIO can set is 0.001.

```
538     SIOERR^EXTSIZE^OVERFLOW: Extent size is greater than
        65535 pages
```

Cause. The CHECK^FILE procedure could not return the primary or secondary extent value because the extent size is greater than 65,535 pages.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Do not attempt the SIO operation listed above.

```
539     SIOERR^PAGEWRITE^OVERFLOW: The highest possible page
        in EDIT file has been written.
```

Cause. While writing the data to the edit file, the SIO procedure call attempted to write a page above the edit file limit.

Effect. Data cannot be written further in the edit file.

Recovery. Use multiple edit files instead of a single edit file to write data or use a structured file, for example, Entry-sequenced.

4 DEFINE Errors

This section contains errors that relate specifically to DEFINE attribute sets. DEFINE errors can occur when you create, save, restore, delete, or otherwise manipulate DEFINEs of any class using the DEFINE procedures.

Error Codes

This subsection lists each DEFINE error code and provides a description of each code.

```
0       (%000000)  Successful completion.
```

Cause. The call completed successfully.

Effect. The operation is successful.

Recovery. Informative message only; no corrective action is needed.

```
2049    (%004001)  A syntax error occurred in the DEFINE
                   name.
```

Cause. There was a syntax error in the DEFINE name.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Correct the syntax error, then reissue the request.

```
2050    (%004002)  The DEFINE name already exists.
```

Cause. The DEFINE name being added already exists.

Effect. The procedure returns the error code and does not add the DEFINE name.

Recovery. Correct or change the DEFINE name, then reissue the request.

```
2051    (%004003)  The DEFINE name does not exist.
```

Cause. The specified DEFINE could not be found.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Check that you have specified the correct DEFINE name, then reissue the request.

```
2052    (%004004)  Unable to obtain file-system buffer
                   space.
```

Cause. Either file-system buffer space was not available or there is not enough room in the process file segment (PFS). The PFS is an extended data segment associated with the process that, among other things, contains DEFINE information. For example, if too many files are open or if many nowait I/O operations are outstanding, there might not be enough space in the PFS.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Close some files, wait for nowait I/O to finish, then try again. For G-series releases earlier than G06, restart with a larger PFS if appropriate.

```
2053    (%004005)  Unable to obtain physical memory.
```

Cause. The physical memory available was not enough to perform the requested operation.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Wait, then try again. If the problem persists, check the system for processes that use too much memory.

```
2054    (%004006)  There was a bounds error in a
                   parameter.
```

Cause. A bounds error occurred in a parameter of the procedure call.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. This is a coding error; corrective action is application-dependent.

```
2055    (%004007)  An attribute is not allowed for
                   the current CLASS.
```

Cause. One of the attributes provided is not allowed for the current DEFINE class.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Correct the attribute for the current class, then retry the operation.

```
2056    (%004010)  An attribute is missing.
```

Cause. An attribute was missing from the procedure call.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Add the attribute, then retry the operation.

```
2057    (%004011)  An attribute required for the
                   current DEFINE CLASS is missing
                   from the working set.
```

Cause. An attribute that is required for this DEFINE class is missing from the working set.

Effect. If the operation is DEFINESAVE, the working set is saved. If the operation is DEFINERESTORE, the saved DEFINE is restored to the working attribute set. If the operation is DEFINEADD, the DEFINE is not added.

Recovery. Add the attribute for the current class, then retry the operation.

```
2058    (%004012)  The working set is inconsistent for
                   the current CLASS of DEFINEs.
```

Cause. The working set was inconsistent (two or more attributes have conflicting values) for the current DEFINE class.

If the consistency check failed for a SORT DEFINE, the consistency check number returned indicates the cause.

Cod e	**Meaning**
001	Use SCRATCHON or NOSCRATCHON, but not both.

If the consistency check failed for a TAPE DEFINE, the consistency check number returned indicates the cause:

Cod e	**Meaning**
001	Use RETENTION or EXPIRATION, not both.
002	USE IN and USE EXTEND require LABELS ANSI, LABELS IBM, or LABELS IBMBACKUP. In addition, if REELS is specified, the value must equal the number of volumes listed for VOLUME.
003	VOLUME is required with LABELS ANSI, LABELS IBM, or LABELS IBMBACKUP.
004	The EBCDIC attribute cannot be used with LABELS ANSI.
005	If RECFORM F is specified, BLOCKLEN must be a multiple of RECLEN.
006	Use DEVICE or SYSTEM, but not both.

Cod e	Meaning
007	DEVICE is required for LABELS BYPASS or LABELS OMITTED. The BLOCKLEN, EBCDIC, EXPIRATION, FILEID, FILESECT, FILESEQ, GEN, OWNER, RECFORM, RECLEN, REELS, RETENTION, SYSTEM, USE, VERSION, and VOLUME attributes cannot be used with LABELS BYPASS or LABELS OMITTED.
008	VOLUME SCRATCH cannot be specified with USE IN or USE EXTEND.
009	FILEID must be specified with LABELS IBM or LABELS IBMBACKUP.
010	RECLEN must be greater than 0 when BLOCKLEN is omitted.

Note. The following CLASS DEFINEs do not return error 2058: CATALOG, DEFAULTS, MAP, SEARCH, and SUBSORT.

If the consistency check failed for a TAPECATALOG DEFINE, the consistency check number returned indicates the cause.

Code	**Meaning**
001	Use RETENTION or EXPIRATION, but not both.
005	If RECFORM F is specified, BLOCKLEN must be a multiple of RECLEN.
006	Use DEVICE or AVRSYSTEM, but not both.
101	USE cannot be used with LABELS OMITTED.
102	USE is required.
103	FILEID is required.
104	CATALOG OFF is required when LABELS BYPASS or LABELS OMITTED is used.
105	COMMENT, OWNER, RETENTION, TAPEMODE, VERSION NEW, and VOLUME SCRATCH cannot be used with USE IN.
106	VOLUME is required with USE EXTEND.
107	DEVICE is required when LABELS BYPASS or LABELS OMITTED is used.
108	VOLUME is required when LABELS ANSI, LABELS BACKUP, LABELS IBM, or LABELS IBMBACKUP is used with USE IN and CATALOG OFF.
109	The character string for OWNER exceeds the maximum length for LABELS IBM or LABELS IBMBACKUP.
110	BLOCKLEN, RECFORM, and RECLEN cannot be used with LABELS BACKUP, LABELS BYPASS, or LABELS OMITTED.
112	EBCDIC can be used only with LABELS IBM or LABELS IBMBACKUP.
113	AVRSYSTEM, COMMENT, FILECAT, FILEID, GEN, OWNER, POOL, RETENTION, USE, VERSION, and VOLCAT cannot be used with LABELS BYPASS.
114	AVRSYSTEM, OWNER, and VOLUME cannot be used with LABELS OMITTED.
115	USE EXTEND cannot be used with LABELS BACKUP or LABELS IBMBACKUP.
116	GEN must be an absolute value when CATALOG OFF is used.
117	GEN cannot be specified as a relative generation when USE OUT is used.
118	GEN cannot be +1 when USE IN is used.

Effect. If the operation is DEFINESAVE, the working set is saved. If the operation is DEFINERESTORE, the saved DEFINE is restored to the working attribute set.

Recovery. Correct the inconsistency, then retry the operation.

```
2059    (%004013)  The working set for DEFINEs is invalid.
```

Cause. The working set is invalid.

Effect. If the operation is DEFINESAVE, the working set is saved. If the operation is DEFINERESTORE, the saved DEFINE is restored to the working attribute set.

Recovery. Correct the problem, then retry the operation.

```
2060    (%004014)  No more DEFINEs are allowed.
```

Cause. The DEFINE name specified in the DEFINENEXTNAME call is the last DEFINE. The *define-name* parameter value remains unchanged.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Informative message only; no corrective action is needed.

```
2061    (%004015)  No more attributes are allowed for
                   the DEFINE.
```

Cause. A call was made to DEFINEREADATTR, but no other attributes exist for the DEFINE after the last one returned. The last call for the DEFINE was successful.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Informative message only; no corrective action is needed.

```
2062    (%004016)  The attribute is invalid for the
                   DEFINE.
```

Cause. There was a syntax error in an attribute name.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Correct the syntax, then reissue the operation.

```
2063    (%004017)  The name of the default subvolume has a
                   syntax error in the SET DEFINE command.
```

Cause. There was a syntax error in a default name.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Correct the name in the default parameter, then reissue the operation.

```
2064    (%004020)  Resetting a required DEFINE attribute
                   is not allowed.
```

Cause. An attempt was made to reset a required DEFINE attribute.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. You cannot reset a required DEFINE attribute.

```
2066    (%004022)  A required parameter is missing.
```

Cause. A required parameter was not supplied.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Add the missing parameter, then reissue the request.

```
2067    (%004023)  An invalid value was supplied for an
                   attribute.
```

Cause. An illegal value was supplied for an attribute.

Effect. The procedure returns the error code and does not perform the requested operation.

Recovery. Correct the value, then reissue the request.

```
2068    (%004024)  The CLASS name identifies a
                   nonexistent CLASS.
```

Cause. The DEFINE class name identified a nonexistent class.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the class name, then reissue the request.

```
2069    (%004025)  Attempt to add a DEFINE that does not
                   fall under current DEFMODE setting.
```

Cause. The process's DEFINE mode (DEFMODE) did not allow the DEFINE to be added.

Effect. The procedure returns the error code and does not add the DEFINE.

Recovery. Correct the DEFMODE setting for the desired operation.

```
2073    (%004031)  Replacing the =_DEFAULTS DEFINE with a
                   DEFINE having the same name but a
                   class other than DEFAULTS is not
                   allowed.
```

Cause. The =_DEFAULTS DEFINE cannot be replaced with a DEFINE having the same name but a class other than DEFAULTS.

Effect. The procedure returns the error code and does not replace the =_DEFAULTS DEFINE.

Recovery. Specify the DEFINE with a DEFAULTS class, then retry the operation.

```
2074    (%004032)  Deleting the DEFINE is not allowed.
```

Cause. Some DEFINEs cannot be deleted. An attempt to delete the =_DEFAULTS DEFINE, for example, results in this message.

Effect. The DEFINE is not deleted.

Recovery. You cannot delete the =_DEFAULTS DEFINE.

```
2075    (%004033)  A DEFINE option is invalid.
```

Cause. DEFINERESTORE *option*.<0:13> must be 0 or DEFINESAVE *option*.<0:14> must be 0.

Effect. The option not selected must be set to 0 even though it is not used. Until it is set to 0, other options cannot be selected.

Recovery. Set the unused option to 0.

```
2076    (%004034)  The buffer is too small for the saved
                   DEFINE.
```

Cause. The buffer was too small to contain the DEFINE.

Effect. The DEFINE is not saved. The buffer size required to save the DEFINE is returned in the *length* parameter.

Recovery. Repeat the DEFINESAVE operation with a buffer at least as large as the length returned from the failed attempt.

```
2077    (%004035)  An extended address parameter referred
                   to an invalid segment.
```

Cause. The DEFINERESTORE *buffer* or *define-name* is in an invalid segment.

Effect. The DEFINERESTORE operation is not performed.

Recovery. Correct the address and retry.

```
2078    (%004036)  The DEFINERESTORE buffer
                   does not contain a valid saved DEFINE.
```

Cause. DEFINERESTORE *buffer* does not contain a valid saved DEFINE created by DEFINESAVE.

Effect. The DEFINE is not added to the current set or to the working set.

Recovery. Recovery is not possible.

```
2079    (%004037)  Cannot save the working set because the
                   name is =_DEFAULTS and the working set
                   is not class DEFAULTS.
```

Cause. The working set could not be saved using DEFINESAVE because the *define-name* is =_DEFAULTS and the working set is not of class DEFAULTS.

Effect. The working set is not saved.

Recovery. To save the named working set, change the class of the working set to class DEFAULTS.

If the DEFAULTS DEFINE is saved, the DEFINERESTORE replace option must be used to restore it as an active DEFINE because the =_DEFAULTS DEFINE is always present.

```
2081     (%004041)   Bad internal format.
```

Cause. Some internal error occurs during DEFINE propagation.

Effect. Process creation fails.

Recovery. Retry the operation. Contact HP support in case of repeated failure.

5 NEWPROCESS AND NEWPROCESSNOWAIT Errors

The following error codes and error lists are produced by the process-control procedures NEWPROCESS and NEWPROCESSNOWAIT. These procedures can only create processes with process identification numbers (PINs) in the low range of 0 through 254.

The NEWPROCESS and NEWPROCESSNOWAIT procedures return errors in a format that is different from other procedures. In many cases, the error message must be decoded. The error returned in the *error* parameter of the NEWPROCESS procedure is explained in the *Guardian Procedure Calls Reference Manual*. The error returned in the rightmost half (bits <8:15>) contains either an error subcode or a file-system error number. Refer to Section 2, File-System Errors, for an explanation of the file-system error number. When file-system error number 119 is returned in bits <0:7> or <8:15>, look in the *errinfo* parameter for an explanation of the error.

On D-series and later releases, you can use the PROCESS_LAUNCH_ or PROCESS_CREATE_ procedures. With these procedures, you can create processes with PINs in the low range (0 through 254) or in the high range (256 through 65535). Because you might want your processes to communicate with processes created with the NEWPROCESS or NEWPROCESSNOWAIT procedure, you might want to specify low PINs for processes that could require such communication.

For information about PROCESS_LAUNCH_ and PROCESS_CREATE_ procedure errors, see Section 6, Process Creation Errors.

For further information about creating processes, refer to the discussions of these procedures in the *Guardian Programmer's Guide* and the *Guardian Procedure Calls Reference Manual*.

Error Codes

A NEWPROCESS error code is returned to the calling procedure as two 8-bit fields or, if returned by the command interpreter or other interactive program, the message is displayed on the terminal as a 6-digit octal number. The leftmost half of the error word (bits <0:7>) contains the NEWPROCESS error number; the rightmost half (bits <8:15>) contains either an error subcode or a file-system error number.

```
0        (%000000)     NO ERROR
```

Cause. The call was completed successfully.

Effect. None.

Recovery. None.

```
1        (%000400)    UNDEFINED EXTERNAL(S)
```

Cause. The process being started contains a call to an external procedure that is not in the operating system code area, the user library (if applicable), or the application code area.

Process creation occurs, and a message is printed on the home terminal. For example:

```
PID: \SYS10.4,94 \SYS10.$XL.SVOL.TEST (TNS)
External References Not Resolved to Any User/System Library:
Prg: \SYS10.$XL.SVOL.TEST -> MY_PROC (PROC)
Undefined externals
```

where MY_PROC is the name of the undefined external procedure in the program file \SYS10.$XL.SVOL.TEST.

Effect. When the operating system finds a call to an undefined external procedure, it replaces the original call with a call to Debug. If the operating system tries to invoke the undefined external during program execution, the program goes into the debug state.

Recovery. Either correct the coding error, use linker (Binder or `nld` utility) to add the procedure to the code area or user library, or refer the call to a procedure that is already present in the application program.

```
2        (%001000)    NO PCB AVAILABLE
```

Cause. All entries in the configured process control block (PCB) table for the processor were in use or the process required a low PIN but none were available.

Effect. No process is created.

Recovery. Try a different processor or wait and then retry the existing call.

```
3        (%001400 + %nnn)  FILE-SYSTEM ERROR ON program-file
```

Cause. The system monitor encountered a file-system error while accessing *program-file* during process creation.

Effect. The call returns the error number of the file-system error (%*nnn*) in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
4         (%002000)      UNABLE TO ALLOCATE MAP
```

Cause. Not enough space was available in the processor's MAPPOOL to permit the system monitor to generate the code and data-map copies required by the new process (a configuration problem).

Effect. No process is created.

Recovery. Try a different processor or wait and then retry the existing call.

```
5         (%002400 + %nnn)    FILE-SYSTEM ERROR ON SWAP FILE
```

Cause. An error occurred during the creation or opening of the swap file.

Effect. File-system error %*nnn* is returned in `error`.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
6         (%003000 + %nnn)   ILLEGAL FILE FORMAT
```

Cause. When the system monitor checked on whether this file is a program, the program or library file failed the test. (The system monitor includes checks for file codes 100 and 700.)

Bits <8:15> (%*nnn*) contain an error subcode that indicates the invalid file format. Bit 15 is set to 0 if a program file is in error and to 1 if a library file is in error. Error subcodes are listed in Table 5-1.

Table 5-1. NEWPROCESS Error 6 Error Subcodes (page 1 of 5)

Decima l	Octal	Meaning
2	%002	The program file is not a disk file.
3	%003	The library file is not a disk file.
4	%004	The program file does not have file code 100 or 700.
5	%005	The library file does not have file code 100 or 700.
6	%006	The program file does not have the correct file structure.
7	%007	The library file does not have the correct file structure.
8	%010	The program file requires a later version of the operating system.
9	%011	The library file requires a later version of the operating system.
10	%012	The program file does not have a main procedure.
13	%015	The library file has a main procedure.
14	%016	Program file has a stack definition of zero pages.
16	%020	The program file has an invalid procedure entry-point (PEP) table.
17	%021	The library file has an invalid procedure entry-point (PEP) table.
18	%022	The initial extended segment information in the program file is inconsistent.
20	%024	The program file resident size is greater than the code area length.
21	%025	The library file resident size is greater than the code area length.
22	%026	The program file was not prepared by the `nld` utility or the Binder program.
23	%027	The library file was not prepared by the `nld` utility or the Binder program.
24	%030	The program file has undefined data blocks.
25	%031	The library file has undefined data blocks.
26	%032	The program file has data blocks with unresolved references.
27	%033	The library file has data blocks with unresolved references.
28	%034	The program file has too many (more than 32) TNS code segments.
29	%035	The library file has too many (more than 32) TNS code segments.
30	%036	TNS/R code length in the program file is invalid.
31	%037	TNS/R code length in the library file is invalid.
32	%040	TNS/R code address in the program file is invalid.
33	%041	TNS/R code address in the library file is invalid.
34	%042	TNS/R data length in the program file is invalid.
35	%043	TNS/R data length in the library file is invalid.

Table 5-1. NEWPROCESS Error 6 Error Subcodes (page 2 of 5)

Decimal	Octal	Meaning
36	%044	TNS/R data address in the program file is invalid.
37	%045	TNS/R data address in the library file is invalid.
38	%046	Program file has too many TNS/R code segments.
39	%047	Library file has too many TNS/R code segments.
40	%050	Program file has invalid TNS/R resident areas.
41	%051	Library file has invalid TNS/R resident areas.
42	%052	Accelerator header in program file is invalid.
43	%053	Accelerator header in library file is invalid.
44	%054	UC (user code) was accelerated with the wrong virtual address.
45	%055	UL (user library) was accelerated with the wrong virtual address.
46	%056	Program file has entry in TNS/R fixup list with invalid external entry-point (XEP) index value or invalid code address value.
47	%057	Library file has entry in TNS/R fixup list with invalid external entry-point (XEP) index value or invalid code address value.
48	%060	Accelerated program file has external procedure identifier list (EPIL), internal procedure identifier list (IPIL), or external entry-point (XEP) table with incorrect format.
49	%061	Accelerated library file has external procedure identifier list (EPIL), internal procedure identifier list (IPIL), or external entry-point (XEP) table with incorrect format.
50	%062	UC (user code) was accelerated using the wrong Accelerator option (UC, UL, SC, or SL).
51	%063	UL (user library) was accelerated using the wrong Accelerator option (UC, UL, SC, or SL).
52	%064	Program file was accelerated with incompatible version of the Accelerator.
53	%065	Library file was accelerated with incompatible version of the Accelerator.
54	%066	Program file has invalid callable gateway (GW) table.
55	%067	Library file has invalid callable gateway (GW) table.
56	%070	Wrong processor type is target in program file.
57	%071	Wrong processor type is target in library file.
58	%072	Program file has inconsistent TNS/R fixup list information.
59	%073	Library file has inconsistent TNS/R fixup list information.
60	%074	An internal structure of the program file contains an error.
61	%075	An internal structure of the library file contains an error.

Table 5-1. NEWPROCESS Error 6 Error Subcodes (page 3 of 5)

Decima l	Octal	Meaning
62	%076	An internal structure of the program file contains an error.
63	%077	An internal structure of the library file contains an error.
64	%100	An internal structure of the program file has an entry point value of 0.
66	%102	An internal structure of the program file contains an error.
67	%103	An internal structure of the library file contains an error.
68	%104	The list of unresolved procedure names in the program file contains an error.
69	%105	The list of unresolved procedure names in the library file contains an error.
70	%106	The fixup computed an invalid file offset to the code area of the program file.
71	%107	The fixup computed an invalid file offset to the code area of the library file.
72	%110	The program file has an invalid fixup item.
73	%111	The library file has an invalid fixup item.
74	%112	An internal structure of the program file contains an error.
75	%113	An internal structure of the library file contains an error.
76	%114	The program file has an instruction at a call site that is not the type expected for its fixup item.
77	%115	The library file has an instruction at a call site that is not the type expected for its fixup item.
78	%116	The header of a TNS/R native program file is not in correct format.
79	%117	The header of a TNS/R native library file is not in correct format.
80	%120	The code in the program file starts at the wrong virtual address.
81	%121	The code in the library file starts at the wrong virtual address.
82	%122	The program file has too much data for the main stack.
84	%124	The code area of the program file is too large.
85	%125	The code area of the library file is too large.
86	%126	The program file has a gateway (GW) table but no callable procedures.
87	%127	The library file has a gateway (GW) table but no callable procedures.
89	%131	The file codes of the program file and library file do not match.
90	%132	The program file being started can run only in the Guardian environment and it is being started in the OSS environment, or vice versa.

Table 5-1. NEWPROCESS Error 6 Error Subcodes (page 4 of 5)

Decimal	Octal	Meaning
91	%133	The library file being started can run only in the Guardian environment and it is being started in the OSS environment, or vice versa.
92	%134	The program and the library conflict on global data mapping. This error is reported on the program. The user library selected is not compatible with the user library specified when the Binder program originally created the program object file.
94	%136	The program needs to import data from the library and the library is not exporting any data. This error is reported on the program.
96	%140	The program file uses a library and is switching to a new library, but the program was accelerated by a version of the Accelerator that cannot handle library data relocation at fixup time.
98	%142	The program file has no code spaces.
99	%143	The library file has no code spaces.
100	%144	The program file is not executable. Either it was not linked with the `nld` utility, it was not linked correctly, or it has been corrupted.
101	%145	The library file is not executable. Either it was not linked with the `nld` utility, it was not linked correctly, or it has been corrupted.
102	%146	The program file is not executable. Either it was linked with an incompatible version of the `nld` utility or it has been corrupted.
103	%147	The library file is not executable. Either it was linked with an incompatible version of the `nld` utility or it has been corrupted.
104	%150	The program file is not executable because it has more than one Tandem information header. An error occurred during the linking of the program file.
105	%151	The library file is not executable because it has more than one Tandem information header. An error occurred during the linking of the library file.
106	%152	The program file is not executable because it has more than one REGINFO information header. An error occurred during the linking of the program file.
107	%153	The library file is not executable because it has more than one REGINFO information header. An error occurred during the linking of the library file.
108	%154	The program file is not executable because it does not have a GINFO information header. An error occurred during the linking of the program file.
109	%155	The library file is not executable because it does not have a GINFO information header. An error occurred during the linking of the library file.

Table 5-1. NEWPROCESS Error 6 Error Subcodes (page 5 of 5)

Decima l	Octal	Meaning
110	%156	The program file is not executable because it does not have either a Tandem information header, a REGINFO information header, or a text header. An error occurred during the linking of the program file.
111	%157	The library file is not executable because it does not have either a Tandem information header, a REGINFO information header, or a text header. An error occurred during the linking of the library file.
112	%160	The program file specifies too many shared run-time libraries (SRLs).
113	%161	The library file specifies too many shared run-time libraries (SRLs).
114	%162	The program file specifies duplicate shared run-time libraries (SRLs).
115	%163	The library file specifies duplicate shared run-time libraries (SRLs).
117	%165	The shared run-time library (SRL) does not export any procedures.
121	%171	The shared run-time library (SRL) does not have a file code of 700.
122	%172	The list of unresolved flags in the program file contains an error.
124	%174	An attempt was made to spawn a shell script on a remote node.

Effect. No process is created.

Recovery. Take corrective action as indicated by the subcode. For example, if subcode 2, 3, 4, or 5 is returned, use the FUP INFO command or the FILEINFO procedure to check the file code.

```
7       (%003400)   UNLICENSED PRIVILEGED PROGRAM
```

Cause. The program file contains procedures having CALLABLE or PRIV attributes, but the program file is not licensed to execute in privileged mode and the super ID was not running the program.

Effect. No process is created.

Recovery. Have the super ID license the program.

```
8        (%004000 + %nnn)    PROCESS NAME ERROR
```

Cause. The process name was invalid.

Effect. File-system error number %*nnn* is returned in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
9        (%004400)    LIBRARY CONFLICT
```

Cause. The call specified a library file, but the program was either already running with a different library or was not running with a library and a library was specified.

Effect. No process is created.

Recovery. All processes running a given program must use the same library. Always specify the library file name to avoid the conflicts that can arise when the library file is modified.

```
10       (%005000 + %nnn)   UNABLE TO COMMUNICATE
                            WITH SYSTEM MONITOR PROCESS
```

Cause. The process could not communicate with the system monitor process, possibly because the processor module where the program was to be run did not exist or was inoperable. %*nnn* in *error*.<8:15> is the file-system error and indicates why the NEWPROCESS error occurred.

Effect. No process is created.

Recovery. Select another processor, then try again.

```
11       (%005400 + %nnn)    FILE-SYSTEM ERROR ON LIBRARY FILE
```

Cause. The system monitor encountered a file-system error while accessing library file during process creation.

Effect. File-system error number %*nnn* is returned in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
12       (%006000)   PROGRAM FILE AND LIBRARY FILE ARE THE SAME
```

Cause. The program file and library file are the same file.

Effect. No process is created.

Recovery. Select a different library file for the program.

```
13       (%006400 + %nnn)   INVALID SEGMENT SIZE UNABLE
                            TO SET UP THE PAGE TABLES
```

Cause. The operating system could not set up an extended data segment for the program.

Effect. File-system error number %*nnn* is returned in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
14       (%007000 + %nnn)   FILE-SYSTEM ERROR ON INITIAL
                            SETUP OF THE SWAP FILE
```

Cause. A file-system error occurred while the operating system was trying to set up the swap file for a COBOL program.

Effect. File-system error number %*nnn* is returned in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
15       (%007400 + %nnn)   ILLEGAL HOME TERMINAL
```

Cause. The home terminal name for the new process does not exist or is not a legal process or terminal name.

Effect. File-system error number%*nnn* is returned in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
16       (%010000 + %nnn)   I/O ERROR ON HOME TERMINAL
```

Cause. An I/O error occurred at the home terminal. There are undefined externals in the object file, and NEWPROCESS cannot OPEN or WRITE to the home terminal to display the undefined-externals message.

Effect. The error number of file-system error %*nnn* is returned in *error*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number %*nnn*.

```
17      (%010400 + %nnn)  DEFINE CONTEXT PROPAGATION
                          ERROR
```

Cause. An error occurred when existing DEFINEs were being propagated.

Bits <8:15> contain a subcode that indicates the cause of the error, as follows:

Subcode	**Meaning**
0	Unable to convert a DEFINE name to network form.
2	Internal error.
3	Illegal DEFMODE supplied.

Effect. No process is created.

Recovery. Retry the call to NEWPROCESS. If errors recur, contact your service provider.

```
18      (%011000)  OBJECT FILE WITH AN ILLEGAL
                   DEVICE SUBTYPE
```

Cause. An attempt was made to run an object file with an invalid device subtype. The device subtype is an attribute stored in each object file. The process created from an object file is assigned the device subtype stored in that object file. Only named processes are allowed nonzero device subtypes.

Device subtypes in the range 1 through 15 are reserved for processes that are:

- Created by the super ID
- Created from licensed object files
- Created from object files owned and provided by the super ID

Effect. No process is created.

Recovery. Verify that the call to NEWPROCESS contains the *name* parameter. If the *name* parameter exists, either recompile or rebind the object file to change the device subtype to an unrestricted value, or contact the super ID.

```
19        (%011400)   PROCESS DEVICE SUBTYPE SPECIFIED
                      IN BACKUP PROCESS ISN'T THE SAME
                      AS PRIMARY'S
```

Cause. The operating system tried to create a process, but the backup process device subtype was not from the same program file as the primary process's device subtype. This error is probably a programming error.

Effect. No process is created.

Recovery. Find out which object file was being passed to NEWPROCESS and correct the error.

```
22        (%013000)   PFS^SIZE IS OUT OF RANGE
```

Cause. The optional parameter *pfs-size* specified a value outside the range of 128K through 1024K bytes.

Effect. No process is created.

Recovery. Correct the specification to a value within the acceptable range. If *pfs-size* is not specified or if 0 is specified, the default size of 256K bytes is used.

```
23        (%013400)   CANNOT CREATE PFS
```

Cause. NEWPROCESS or NEWPROCESSNOWAIT cannot allocate the process file segment (PFS).

Effect. No process is created.

Recovery. If insufficient PFS space was the cause of this error, either try to free up more PFS space or wait for more PFS space to become available.

```
24        (%014000 + %nnn)   UNKNOWN ERROR
```

Cause. An error was received when an attempt was made to start a process on a remote system. Your system does not know how to interpret the error.

Effect. The call returns the PROCESS_CREATE_ error number %*nnn*, or 119 if the actual error number is too large, in *errinfo*.<8:15>. No process is created.

Recovery. Refer to Section 6, Process Creation Errors, for corrective action for the PROCESS_CREATE_ error received in bits <8:15> of the *errinfo* parameter, and then perform a recovery action based on this error. If the problem persists and the PROCESS_CREATE_ error number does not give you enough information to diagnose the problem, contact your service provider.

```
25        (%014400)   UNABLE TO ALLOCATE A PRIV STACK
                      FOR THE PROCESS
```

Cause. There was no segment available for the priv stack of a TNS/R native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
26        (%015000)   UNABLE TO LOCK THE PRIV STACK
                      FOR THE PROCESS
```

Cause. There was not enough physical memory free to lock the priv stack of a TNS/R native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
27        (%015400)   UNABLE TO ALLOCATE A MAIN STACK
                      FOR THE PROCESS
```

Cause. There was no segment available for the main stack of a TNS/R native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
28        (%016000)   UNABLE TO LOCK THE MAINSTACK OF A TNS/R
                      NATIVE I/O PROCESS THAT IS CREATED BY
COUP
```

Cause. There was not enough physical memory free to lock the main stack for a TNS/R native I/O process. This error is only returned to some privileged callers on D-series releases.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
29        (%016400 + %nnn)   SECURITY INHERITANCE FAILURE
```

Cause. An error occurred during an attempt to obtain or propagate security identity information.

Effect. The call returns the file-system error number `%`*nnn* in *errinfo*.<8:15>. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in *errinfo*.<8:15>. If the file-system error indicates an invalid operation or an invalid parameter, the problem might be caused by a version mismatch between the NonStop operating system and Standard Security. Correct your system configuration if this is the case; otherwise, the problem is likely to be an internal error that should be reported to your service provider.

```
30        (%017000)  UNABLE TO ALLOCATE NATIVE GLOBALS
```

Cause. The system was unable to allocate the global data segment of a native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
31        (%017400)  UNABLE TO LOCK NATIVE GLOBALS
```

Cause. The system was unable to lock the global data segments of a native I/O process. This error is only returned to some privileged callers.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
32        (%020000)  MAIN STACK MAXIMUM VALUE TOO LARGE
```

Cause. The main stack maximum value is too large.

Effect. No process is created.

Recovery. Use the `nld` utility to specify a new value that is within the valid range.

```
33        (%020400)  HEAP MAXIMUM VALUE TOO LARGE
```

Cause. The heap maximum value is too large.

Effect. No process is created.

Recovery. Use the `nld` utility to specify a new value that is within the valid range.

```
34        (%021000)  SPACE GUARANTEE VALUE TOO LARGE
```

Cause. The space guarantee value is too large.

Effect. No process is created.

Recovery. Use the `nld` utility to specify a new value that is within the valid range.

```
35        (%021400 + %nnn)  DUPLICATE SRL
```

Cause. The process creation request specifies duplicate shared run-time libraries (SRLs).

Effect. `%`*nnn* in *error*.<8:15> contains the SRL numbers of the duplicate SRLs in the form *xxyy* (where *xx* is the first SRL and *yy* is the duplicate SRL). No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
36        (%022000 + %nnn)  UNKNOWN SRL SPECIFIED BY PROGRAM
```

Cause. The system was unable to find a shared run-time library (SRL) specified by the program file.

Effect. No process is created. `%`*nnn* in *error*.<8:15> contains the SRL number of the SRL that could not be found.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
37        (%022400 + %nnn)  UNKNOWN SRL SPECIFIED BY ANOTHER
SRL
```

Cause. The system was unable to find a shared run-time library (SRL) specified by another SRL.

Effect. `%`*nnn* in *error*.<8:15> contains the SRL numbers of the two SRLs in the form *xxyy* (where *xx* is the SRL that specifies the *yy* SRL). No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
38        (%023000 + %nnn)  TOO MANY SRLS
```

Cause. The process creation request specifies too many shared run-time libraries (SRLs).

Effect. `%`*nnn* in *error*.<8:15> contains the maximum number of SRLs that can be specified. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file, using no more than 32 SRLs, the maximum number of SRLs permitted.

```
39        (%023400 + %nnn)  PROGRAM FILE REQUIRES FIXUP
```

Cause. The program file requires fixups to a shared run-time library (SRL) that is unavailable because it is running.

Effect. `%`*nnn* in *error*.<8:15> contains the SRL number of the running SRL. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
40        (%024000 + %nnn)  SRL REQUIRES FIXUPS
```

Cause. A shared run-time library (SRL) requires fixups to another SRL that is unavailable because it is running.

Effect. `%nnn` in *error*.<8:15> contains the SRL numbers of the two SRLs in the form *xxyy* (where *xx* is the SRL that requires the fixup to the running *yy* SRL). No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
41        (%024400 + %nnn)  PROGRAM FILE SECURITY VIOLATION
```

Cause. A security violation occurred. The program file is not licensed but a shared run-time library (SRL) is licensed and has instance data.

Effect. `%nnn` in *error*.<8:15> contains the SRL number of the licensed SRL. No process is created.

Recovery. License the program file, if possible.

```
42        (%025000 + %nnn)  SRL SECURITY VIOLATION
```

Cause. A security violation occurred. Either the program file or a shared run-time library (SRL) is licensed, but another SRL is not licensed.

Effect. `%nnn` in *error*.<8:15> contains the SRL number of the unlicensed SRL. No process is created.

Recovery. Either remove the licensing of the program, if possible, or use the `nld` utility to specify another SRL.

```
43        (%025400 + %nnn)  PROGRAM REQUIRES SYMBOL FROM SRL
```

Cause. The program file requires a symbol from a shared run-time library (SRL) but the SRL is not exporting it.

Effect. `%nnn` in *error*.<8:15> contains the SRL number of the SRL that does not export the required symbol. No process is created.

Recovery. Use the `nld` utility to specify another SRL that contains the desired data block.

```
47        (%027400)  CANNOT GUARANTEE SWAP SPACE
```

Cause. Requested swap space for the TNS/R native process cannot be guaranteed.

Effect. No process is created.

Recovery. Either use the `nld` utility to shrink the guaranteed swap space specification, use NSKCOM to add additional swap space for the target processor, or run on another processor.

```
48        (%030000 + %nnn)  MISMATCH ON NUMBER OF SRLS
```

Cause. The number of shared run-time libraries (SRLs) specified by an SRL is incorrect.

Effect. `%`*`nnn`* in *`error`*.<8:15> contains the SRL number of the SRL that caused the error. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
49        (%030400 + %nnn)  UNDEFINED EXTERNALS IN SRL
```

Cause. A shared run-time library (SRL) has undefined externals.

Effect. `%`*`nnn`* in *`error`*.<8:15> contains the SRL number of the SRL that has undefined externals. No process is created.

Recovery. Check that the SRL versions match and obtain matching versions if they don’t. If the problem persists, contact your service provider.

```
50        (%031000)  INCORRECT NUMBER OF SRLS IN PROGRAM FILE
```

Cause. The number of shared run-time libraries (SRLs) specified for the program file is incorrect.

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file.

```
51      (%031400)  INCORRECT NUMBER OF SRLS IN LIBRARY
```

Cause. The number of shared run-time libraries (SRLs) specified for the library file is incorrect.

Effect. No process is created, and a message is printed on the home terminal. For example:

```
PID: \SAT.0,36 \SAT.$DATA.MDTUNRES.TC03 (ELF)
SRL-Client References Symbols Not Found In Nominated SRL:
Prg: \SAT.$DATA.MDTUNRES.TC03E -> FRED_PROC (PROC) -> SRL:
     \$DATA.MDTUNRES.SRL1
*ERROR* PROCESS_CREATE_ Error: 51,0
```

Recovery. Use the original object files and the `nld` utility to re-create the program file.

```
52       (%032000)  SRL MUST BE LICENSED
```

Cause. A security violation occurred. A shared run-time library (SRL) must be licensed to be used by callable or privileged code.

Effect. No process is created.

Recovery. Rebuild the program file. If the error recurs, contact your service provider.

```
53       (%032001)  UNABLE TO OBTAIN GLOBAL VIRTUAL SPACE
```

Cause. Either there is insufficient virtual memory or there is insufficient lockable memory in the processor.

Effect. No process is created.

Recovery. Try again or select another processor. If the problem persists, run the PEEK program in the processor where process creation failed and determine new memory needs.

```
54       (%033000)  SYMBOLIC REFERENCE TARGET/SOURCE TYPE
                    MISMATCH
```

Cause. An attempt was made to resolve a procedure address to a data area or a data address to a procedure area.

Effect. No process is created, and a message is printed on the home terminal. For example:

```
PID: \SAT.1,284 \SAT.$DATA.SRLVOL.NULCLIE2 (ELF)
Native UL Symbolic Reference Error: Target/Source Type Mismatch:
Prg: \SAT.$DATA.SRLVOL.NULCLIE2 -> X_INT (DATA)
*ERROR* PROCESS_CREATE_ Error: 54
```

Recovery. Use the original files and the `nld` utility to recreate the program file.

```
55        (%033001)   EXTERNAL DATA REFERENCE NOT RESOLVED TO
                      ANY USER/SYSTEM LIBRARY
```

Cause. An anonymous data symbol reference is not located in a UL, Native UL, or system library.

Effect. No process is created, and a message is printed on the home terminal. For example:

```
PID: \SAT.1,284 \SAT.$DATA.SRLVOL.NULCLIE2 (ELF)
External References Not Resolved to Any User/System Library:
Prg: \SAT.$DATA.SRLVOL.NULCLIE2 -> X_INT (DATA)
*ERROR* PROCESS_CREATE_ Error: 55
```

Recovery. Either correct the coding error, use `nld` utility to add the data to the data area or user library, or refer the call to the data that is already present in the application program.

```
119       (%073400)   ERROR NUMBER RETURNED IS TOO LARGE
```

Cause. The error returned in *error*.<0:7> is too large to fit into one byte.

Effect. No process is created.

Recovery. Instead of the *error* parameter, specify the *errinfo* parameter, which is a two-word parameter, to obtain complete error information.

```
3xx       SRL HAS INVALID FILE FORMAT
```

Cause. The call contained an invalid file format on shared run-time library (SRL) number *xx*. This error is returned in the *errinfo* parameter, the second word of which contains the error subcode. See Table 6-2, Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3xx, on page 6-7 for possible subcode values.

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
5xx       FILE-SYSTEM ERROR ON SHARED RUN-TIME LIBRARY (SRL)
          NUMBER XX.
```

Cause. There was a file-system error on shared run-time library (SRL) number *xx*. This error is returned in the *errinfo* parameter, the second word of which contains the error subcode.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for error number *errinfo* parameter. If the problem persists, contact your service provider.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a NEWPROCESS or NEWPROCESSNOWAIT error list in a response. HP subsystems return such an error list when, in performing your request, they call the NEWPROCESS or NEWPROCESSNOWAIT procedure directly or indirectly and an error occurs on the call.

The contents of the error list depend on which procedure was called. The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every NEWPROCESS and NEWPROCESSNOWAIT error list, identifies the procedure: its value is either ZGRD-VAL-NEWPROCESS (3) or ZGRD-VAL-NEWPROCESSNOWAIT (4).

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list could include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a NEWPROCESS or NEWPROCESSNOWAIT error list:

- Include all unconditional tokens listed in the error list description.
- Optionally include any or none of the conditional tokens listed in each error-list description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about tokens token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

3: ZGRD-VAL-NEWPROCESS

There was an error during a call to NEWPROCESS.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-OBJECTFILE        token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-LIBRARYFILE       token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-SWAPFILE          token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-PRIORITY          token-type ZSPI-TYP-UINT.
   ZGRD-TKN-MEMORYPAGES       token-type ZSPI-TYP-BYTE.
   ZGRD-TKN-CPU               token-type ZSPI-TYP-BYTE.
   ZGRD-TKN-CRTPID            token-type ZSPI-TYP-CRTPID.
   ZGRD-TKN-PROCESSNAME       token-type ZSPI-TYP-CRTPID.
   ZGRD-TKN-HOMETERM          token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-INSPECTFLAG       token-type ZSPI-TYP-ENAME.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the 16-bit error code returned in the *error* parameter of NEWPROCESS. This error code indicates the outcome of the process-creation attempt.

ZSPI-TKN-PROC-ERR is the procedure code. Its value is ZGRD-VAL-NEWPROCESS (3).

Conditional Tokens

ZGRD-TKN-OBJECTFILE is the file name of the reporting program in internal format.

ZGRD-TKN-LIBRARYFILE is the library file name in internal format.

ZGRD-TKN-SWAPFILE is the data swap-file name in internal format, which is passed for informational purposes only. This swap file is not used. Processes swap to a file that is managed by the Kernel-Managed Swap Facility.

ZGRD-TKN-PRIORITY is the priority of the new process.

ZGRD-TKN-MEMORYPAGES is the size of the data stack in pages.

ZGRD-TKN-CPU is the processor number for the new process.

ZGRD-TKN-CRTPID is the process ID returned by NEWPROCESS. If ZGRD-TKN-CRTPID is 0, no process was created.

ZGRD-TKN-PROCESSNAME is the new process name.

ZGRD-TKN-HOMETERM is the name of the home terminal for the new process.

ZGRD-TKN-INSPECTFLAG contains the selected debugging attributes.

Effect

The effect of the error depends on the NEWPROCESS error code returned.

Recovery

Follow the recovery procedure for the returned NEWPROCESS error code as described earlier in this section.

4: ZGRD-VAL-NEWPROCESSNOWAIT

There was an error during a call to NEWPROCESSNOWAIT.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-OBJECTFILE       token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-LIBRARYFILE      token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-SWAPFILE         token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-PRIORITY         token-type ZSPI-TYP-UINT.
   ZGRD-TKN-MEMORYPAGES      token-type ZSPI-TYP-BYTE.
   ZGRD-TKN-CPU              token-type ZSPI-TYP-BYTE.
   ZGRD-TKN-PROCESSNAME      token-type ZSPI-TYP-CRTPID.
   ZGRD-TKN-HOMETERM         token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-INSPECTFLAG      token-type ZSPI-TYP-ENUM.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the 16-bit error code returned in the *error* parameter of NEWPROCESSNOWAIT. This error code indicates the initial outcome of the process-creation attempt.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZGRD-VAL-NEWPROCESSNOWAIT (4).

Conditional Tokens

ZGRD-TKN-OBJECTFILE is the file name of the reporting program in internal format.

ZGRD-TKN-LIBRARYFILE is the library file name in internal format.

ZGRD-TKN-SWAPFILE is the data swap-file name in internal format, which is passed for informational purposes only. This swap file is not used. Processes swap to a file that is managed by the Kernel-Managed Swap Facility.

ZGRD-TKN-PRIORITY is the priority of the new process.

ZGRD-TKN-MEMORYPAGES is the size of the data stack in pages.

ZGRD-TKN-CPU is the processor number for the new process.

ZGRD-TKN-PROCESSNAME is the new process name.

ZGRD-TKN-HOMETERM is the name of the home terminal for the new process.

ZGRD-TKN-INSPECTFLAG contains the selected debugging attributes.

Effect

Effect depends on the returned NEWPROCESSNOWAIT error code.

Recovery

Follow the recovery procedure for the returned NEWPROCESSNOWAIT error code as described earlier in this section.

6 Process Creation Errors

The following error codes and error lists are produced by the PROCESS_LAUNCH_ and PROCESS_CREATE_ procedures, which create a Guardian process. Most of the error codes are also returned by the PROCESS_SPAWN_ procedure, which creates an Open System Services (OSS) process. See Section 9, PROCESS_SPAWN_ Open System Services (OSS) Errors, for more information on error codes and error lists produced by PROCESS_SPAWN_.

You can create a process with a number of different procedures. On C-series releases, you typically used the NEWPROCESS or NEWPROCESSNOWAIT procedure. On D-series and G-series releases, you probably use the PROCESS_LAUNCH_ or PROCESS_CREATE_ procedure. With the PROCESS_LAUNCH_ and PROCESS_CREATE_ procedures, you can create processes with process identification numbers (PINs) in the low range (0 through 254) or in the high range (256 through 65535). In the event that you might want your processes to communicate with processes created with the NEWPROCESS or NEWPROCESSNOWAIT procedure, you might want to specify low PINs for processes that could require such communication.

If PROCESS_LAUNCH_ or PROCESS_CREATE_ is called in a nowait manner, error information is returned in system message -102 (nowait PROCESS_LAUNCH_ or PROCESS_CREATE_ completion). See Section 20, System Messages, for details.

For further information about starting processes and about the PROCESS_LAUNCH_ and PROCESS_CREATE_ procedures, refer to the *Guardian Programmer's Guide* and the *Guardian Procedure Calls Reference Manual*.

Error Codes

The PROCESS_LAUNCH_ and PROCESS_CREATE_ error codes are the values returned to the calling process in the *error* parameter of PROCESS_LAUNCH_ and PROCESS_CREATE_. For many classes of errors, additional information is returned in the *error-detail* parameter.

The PROCESS_SPAWN_ Guardian error codes are the values returned to the calling process in the ZSYS-DDL-PROCESSRESULTS.Z-TPCERROR field of the *process-results* parameter of PROCESS_SPAWN_. For many classes of errors, additional information is returned in the ZSYS-DDL-PROCESSRESULTS.Z-TPCDETAIL field of the *process-results* parameter.

In descriptions of errors that can be returned by PROCESS_LAUNCH_, PROCESS_CREATE_, and PROCESS_SPAWN_, the term “error detail” refers to the information returned in either the *error-detail* parameter of PROCESS_LAUNCH_ and PROCESS_CREATE_ or the ZSYS-DDL-PROCESSRESULTS.Z-TPCDETAIL field of the *process-results* parameter of PROCESS_SPAWN_.

See Section 9, PROCESS_SPAWN_ Open System Services (OSS) Errors, for PROCESS_SPAWN_ error lists and for OSS error codes returned in the ZSYS-DDL-PROCESSRESULTS.Z-ERRNO field of the *process-results* parameter to PROCESS_SPAWN_.

```
0      NO ERROR
```

Cause. The call was completed successfully.

Effect. The process was created, or creation was initiated if the procedure was called in a nowait manner. In the latter case, creation results are returned in a user-level system message.

Recovery. Informative message only; no corrective action is needed.

```
1      A FILE SYSTEM ERROR WAS ENCOUNTERED ON THE PROGRAM FILE
```

Cause. A file-system error occurred on the program file during process creation. The error-detail contains a file-system error number.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
2      PARAMETER ERROR
```

Cause. The call contained an invalid combination of options. For the PROCESS_LAUNCH_ and PROCESS_SPAWN_ procedures, the error-detail information indicates the first parameter to be found in error. See Table 6-1 for possible values. For the PROCESS_CREATE_ procedure, the error-detail information contains the number of first parameter found to be in error, where 1 designates the leftmost parameter.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Check the call to make sure the options are correct and in the proper order.

```
3      BOUNDS VIOLATION
```

Cause. There was a bounds violation on a reference parameter. For the PROCESS_LAUNCH_ and PROCESS_SPAWN_ procedures, the error-detail information indicates the first parameter to be found in error. See Table 6-1 for possible values. For the PROCESS_CREATE_ procedure, the error-detail information contains the number of first parameter found to be in error, where 1 designates the leftmost parameter.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Pass a correct reference address.

Table 6-1. Error Subcodes for PROCESS_LAUNCH_ and PROCESS_SPAWN_ Errors 2 and 3 (page 1 of 2)

error-detail	**PROCESS_LAUNCH_ Structure Field or Parameter in Error**	**PROCESS_SPAWN_ Structure Field or Parameter in Error**
1	Z^PROGRAM^NAME	*oss-program-file*
2	Z^LIBRARY^NAME	Z^LIBRARYNAME
3	Z^SWAPFILE^NAME	Z^SWAPFILENAME
4	Z^EXTSWAPFILE^NAME	Z^EXTSWAPFILENAME
5	Z^PRIORITY	Z^PRIORITY
6	Z^CPU	Z^CPU
9	Z^NAME^OPTIONS	Z^NAMEOPTIONS
10	Z^PROCESS^NAME	Z^PROCESSNAME
14	Z^HOMETERM^NAME	Z^HOMETERM
15	Z^MEMORY^PAGES	Z^MEMORYPAGES
16	Z^JOBID	Z^JOBID
17	Z^CREATE^OPTIONS	Z^CREATEOPTIONS
18	Z^DEFINES^NAME	Z^DEFINES
19	Z^DEBUG^OPTIONS	Z^DEBUGOPTIONS
20	Z^PFS^SIZE	Z^PFSSIZE
21	The program image file is in error. This error subcode is returned if PROCESS_LAUNCH_ is called from COUP; it applies only to some privileged callers on D-series releases.	The program image file is in error. This error subcode is returned if PROCESS_SPAWN_ is called from COUP; it applies only to some privileged callers on D-series releases.
22	*param-list*	Not returned by PROCESS_SPAWN_
23	*error-detail*	Z^TPCDETAIL
24	*output-list*	Not returned by PROCESS_SPAWN_

Table 6-1. Error Subcodes for PROCESS_LAUNCH_ and PROCESS_SPAWN_ Errors 2 and 3 (page 2 of 2)

error-detail	**PROCESS_LAUNCH_ Structure Field or Parameter in Error**	**PROCESS_SPAWN_ Structure Field or Parameter in Error**
25	*output-list-len*	Not returned by PROCESS_SPAWN_
50	Not returned by PROCESS_LAUNCH_	*process-extension*
51	Not returned by PROCESS_LAUNCH_	Z^OSSOPTIONS
52	Not returned by PROCESS_LAUNCH_	*argv*
53	Not returned by PROCESS_LAUNCH_	*envp*
54	Not returned by PROCESS_LAUNCH_	*envp* contains an invalid address.
56	Not returned by PROCESS_LAUNCH_	*inheritance*
57	Not returned by PROCESS_LAUNCH_	An internal error
58	Not returned by PROCESS_LAUNCH_	*fdinfo*
59	Not returned by PROCESS_LAUNCH_	*path*
60	Not returned by PROCESS_LAUNCH_	*inheritance-length*

```
4     A FILE SYSTEM ERROR WAS ENCOUNTERED ON THE USER LIBRARY
FILE
```

Cause. A file-system error occurred on the user library file during process creation. The error-detail contains a file-system error number.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
5     ERROR WHILE ACCESSING SWAP FILE
```

Cause. An error occurred during the creation or opening of the swap file. The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without creating the process.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
6     ERROR WHILE ACCESSING EXTENDED SWAP FILE
```

Cause. An error occurred during the creation, opening, or initial setup of the extended swap file. The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without creating the process.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
7     FILE SYSTEM ERROR ON PFS
```

Cause. An error occurred during the creation of the process file segment (PFS). The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
8     ILLEGAL HOME TERMINAL
```

Cause. The home terminal name for the new process does not exist or is not a legal process or terminal name. The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without creating the process.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
9     I/O ERROR ON HOME TERMINAL
```

Cause. An I/O error occurred at the home terminal. Undefined externals in the program file prevent the operating system from opening or writing to the home terminal to display the undefined-externals message. The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without creating the process.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
10    UNABLE TO COMMUNICATE WITH SYSTEM MONITOR PROCESS
```

Cause. The process could not communicate with the system monitor process, possibly because the processor module where the program was to be run did not exist or was inoperable. The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without creating the process.

Recovery. Select another processor, then try again.

```
11    PROCESS NAME ERROR
```

Cause. The process name was invalid. The error-detail information contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without creating the process.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
12    ILLEGAL PROGRAM FILE FORMAT
```

Cause. An error was detected in the file format of the Program file. The error-detail information contains an error subcode that indicates the invalid file format error. These errors are listed in Table 6-2.

Effect. No process is created.

Recovery. Take corrective action as indicated by the subcode. For example, if subcode 1 or 2 is returned, use the FUP INFO command to check the file code. In many cases, the object file was built improperly or corrupted. It should be relinked and perhaps recompiled.

```
13    ILLEGAL USER-LIBRARY FILE FORMAT
```

Cause. An error was detected in the file format of the user library file. The error-detail information contains an error subcode that indicates the invalid file format error. These error subcodes are listed in Table 6-2.

Effect. No process is created.

Recovery. Take corrective action as indicated by the subcode. For example, if subcode 1 or 2 is returned, use the FUP INFO command to check the file code. In many cases, the object file was built improperly or corrupted. It should be relinked and perhaps recompiled.

Table 6-2. Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3*xx* (page 1 of 4)

Subcode	Meaning
1	The file is not a disk file.
2	For the program file designated in the process creation request: The file is in the Guardian file system and does not have file code 100 or 700, or it is in the OSS file system and not recognizable as a TNS or ELF object file or a shell script. Otherwise: A TNS library file was expected, but the file is in the Guardian file system and does not have file code 100, or the file is in the OSS file system and not recognizable as a TNS object file.
3	The file does not have the correct file structure.
4	The file requires a later version of the operating system.
5	A program lacks an entry point, or an attempt was made to load a library as a program. (An entry point is specified by a TAL or pTAL procedure having the MAIN attribute, or by naming a native procedure in the –e option to the linker.)
6	An attempt was made to load a program as a library, or a TNS user library has a MAIN procedure.
7	A TNS program file does not have data pages.
8	A native object file requires fixup to SRLs by the nld utility, or a TNS object file was not prepared by the Binder program.
9	The file header INITSEGS is not consistent with its size.
10	The file resident size is greater than the code area length.
11	The file was not prepared by the `nld` utility or the Binder program.
12	The file has undefined data blocks.
13	The file has data blocks with unresolved references.
14	The file has too many TNS code segments.
15	Accelerated code length in the file is invalid.
16	Accelerated code address in the file is invalid.
17	Accelerated data length in the file is invalid.
18	Accelerated data address in the file is invalid.
19	The file has too many accelerated code segments.
20	The file has invalid resident areas in accelerated code.
21	Accelerator header in the file is invalid.
22	Either the UC (user code) option or the UL (user library) option was not used when the file was accelerated.
23	File has entry in TNS/R fixup list with invalid external entry-point (XEP) index value or invalid code address value.
24	Accelerated file has external procedure identifier list (EPIL), internal procedure identifier list (IPIL), or external entry-point (XEP) table with incorrect format.

Table 6-2. Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3*xx* (page 2 of 4)

Subcode	Meaning
25	UC (user code) or UL (user library) was accelerated using the wrong Accelerator option (UC, UL, SC, or SL).
26	The file was accelerated with an incompatible version of the Accelerator.
27	The file has an invalid callable gateway (GW) table.
28	The program file contains processor-specific code that cannot be run on the current processor.
29	Fixup of accelerated code was attempted in an object file that was not accelerated.
30	An internal structure of the file contains an error.
31	An internal structure of the file contains an error.
32	An internal structure of the file has an entry point value of 0.
33	An internal structure of the file contains an error.
34	The list of unresolved procedure names contains an error.
35	The fixup computed an invalid file offset to the code area.
36	The file has an invalid fixup item.
37	An internal structure of the file contains an error.
38	The instruction at a call site is not the type expected for its fixup item.
39	Not used.
40	A virtual address specified in an ELF file is outside its allowed range. For example, a text or data segment is specified at an address not valid for this type of file.
41	Not used.
42	The code area or data area is too large.
43	The file either has a gateway (GW) table but no callable procedures or has gateways that are not in the (GW) area.
44	The file codes of the program file and library file do not match. (Not used since G06.12; see 5 and 6 instead.)
45	The file being started can run only in the Guardian environment and it is being started in the OSS environment, or vice versa.
46	Either the TNS program or the TNS User library (but not both) expected the library to contain global data.
47	Either the TNS program needs to import data from the TNS User library and the library is not exporting any data, or the library needs global data space and the program is not providing it.

Table 6-2. Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3*xx* (page 3 of 4)

Subcode	Meaning
48	A TNS program file uses a TNS shared run-time library and is switching to a new library, but the program was accelerated by an old version of the Accelerator that does not support library data relocation at fixup time. Use a version of the Accelerator provided with the D30.00 or later release of the operating system.
49	A TNS object file has no code space.
50	The native object file is not loadable. Either it is a linkable file (such as unlinked compiler output) or it is an incorrect type of loadable file (such as a DLL encountered with a non-PIC program.)
51	The program or library file does not have a valid ELF header for execution on this NonStop operating system. The file either is not targeted for this system, is not an ELF file, or has been corrupted.
52	The file is not executable because it has more than one instance of a header structure unique to HP NonStop Server systems. An error occurred during the linking of the file.
53	An ELF file has a header specifying more than one instance of a segment that should be unique. The file is corrupt or was not built by a valid linker.
54	The non-PIC ELF file is not loadable because it does not have a GINFO header. An error occurred during the linking of the file, or it is corrupt.
55	An ELF file is lacking a required segment.
56	The file specifies too many shared run-time libraries (SRLs).
57	The file specifies duplicate shared run-time libraries (SRLs).
58	The shared run-time library (SRL) does not export any procedures.
60	An ELF library file was expected, but the file is in the Guardian file system and does not have a file code of 700 or it is in the OSS file system and not recognizable as an ELF file.
61	Two related structures in the ELF file have inconsistent lengths.
62	An attempt was made to spawn a shell script on a remote node.
63	An inconsistency exists in the set of public SRLs.
64	Some value (other than an address) specified in an ELF file is outside its range. The file may be corrupted.
65	The current export-digest index specified in an ELF SRL file is greater than the count of export digests in that file. The file is probably corrupted.
66	The count of export digests in an ELF SRL exceeds 256. The file is probably corrupted.
67	A public SRL is marked to require a PIN less than 255; a PIN less than 255 is not allowed.
68	One of the headers that is expected to be at the front of an ELF file did not fit near enough to the front.

Table 6-2. Error Subcodes for Process Creation Errors 12, 13, 70, 76, 84, and 3*xx* (page 4 of 4)

Subcode	Meaning
69	This PIC ELF file is not supported on TNS/R systems: it is licensed, or it contains *callable* functions.
70	The ELF file is too big (EOF > 2^{31} bytes).
71	Some value in the TNS object file header is out of range; the file may be corrupt.
72	The EF_TANDEM_INSTANCE_DATA value in the ELF header is not valid or not consistent with the data program headers found; the file may be corrupt.
73	The p_flags in the ELF header for the resident text header are not as expected; the file may be corrupt.
74	The loadfile has resident text, but no data constant segment and is not marked data_resident. This is not supported.
75	The DLL has callable functions but also has unprotected data. This is not supported.
76	An address to be stored into a relocation site does not fit in 32 bits.
77	The loadfile uses the 64 bit data model. It is not supported on this system.
78	The loadfile is an import library or implicit DLL, not a program, ordinary DLL, or public DLL.

```
14    UNDEFINED EXTERNALS
```

Cause. The process being started contains a call to an external procedure that is not in the operating system code area, the user library (if applicable), or the application code area.

Effect. Process creation occurs, and a message is printed on the home terminal. For example:

```
PID: \SYS10.4,94 \SYS10.$XL.SVOL.TEST (TNS)
External References Not Resolved to Any User/System Library:
Prg: \SYS10.$XL.SVOL.TEST -> MY_PROC (PROC)
Undefined externals
```

where MY_PROC is the name of the undefined external procedure in the program file \SYS10.$XL.SVOL.TEST.

When the operating system finds a call to an undefined external procedure, it replaces the original call with a call to the PROCESS_DEBUG_ procedure. If the process tries to invoke the undefined external during program execution, the program goes into the debug state.

Recovery. Either correct the coding error, use Binder to add the procedure to the code area or user library, or refer the call to a procedure that is already present in the application program.

```
15    NO PCB AVAILABLE
```

Cause. All entries in the process control block (PCB) table for the processor were in use or the process required a low process identification number (PIN) and none were available.

Effect. No process is created.

Recovery. Try a different processor or wait and then retry the existing call.

```
16    UNABLE TO ALLOCATE MAP
```

Cause. Not enough space was available in the processor’s MAPPOOL to permit the system monitor to generate the code and data-map copies required by the new process (this might be a configuration problem).

Effect. No process is created.

Recovery. Too many processes might have been configured to run in the current processor. Try a different processor or wait and then retry the existing call.

```
17    UNLICENSED PRIVILEGED PROGRAM OR LIBRARY
```

Cause. The program file contains procedures having CALLABLE or PRIV attributes, but the program file is not licensed to execute in privileged mode and the super ID was not running the program.

Effect. No process is created.

Recovery. Have the super ID license or run the program under a locally validated super ID.

```
18    LIBRARY CONFLICT
```

Cause. The call specified a library file, but the program was either already running with a different library or was not running with a library.

This error is also generated if a user library file is specified when running an old TNS program containing an “implicit user library.” (Prior to D30, a large TNS program file could be created with 16 segments of user code and up to 16 additional segments mapped as user library. Subsequently, the user-code limit and user-library limits were raised to 32 segments each, and the binder stopped creating programs with an implicit user library.)

Effect. No process is created.

Recovery. All non-PIC processes running a given program must use the same library. Always specify the library file name to avoid the conflicts that can arise when the library file is modified.

```
19    PROGRAM FILE AND LIBRARY FILE ARE THE SAME
```

Cause. The program file and library file are the same file.

Effect. No process is created.

Recovery. Select a different library file for the program.

```
20    PROGRAM FILE WITH AN ILLEGAL DEVICE SUBTYPE
```

Cause. An attempt was made to run a program file with an invalid device subtype. The device subtype is an attribute stored in each program file. The process created from a program file is assigned the device subtype stored in that program file. Only named processes are allowed nonzero device subtypes.

Device subtypes in the range 1 through 15 are reserved for processes that are either:

- Created by the super ID
- Created from licensed program files

- Created from program files owned and provided by the super ID

Subtypes 1 through 47 are reserved for HP use.

Effect. No process is created.

Recovery. Verify either that the call specifies a process name or that the program file is marked as “must run named” by the Binder, the TAL RUNNAMED command, or the C RUNNAMED pragma. If the call specifies a process name, either recompile or rebind the program file to change the device subtype to an unrestricted value, rebind the program file to run named, or contact the super ID.

```
21    PROCESS DEVICE SUBTYPE SPECIFIED IN BACKUP PROCESS
      IS NOT THE SAME AS PRIMARY
```

Cause. The operating system tried to create a process, but the backup process device subtype was not from the same program file as the primary process device subtype. This error is probably a programming error.

Effect. No process is created.

Recovery. Find out which program file was being passed to PROCESS_LAUNCH_, PROCESS_CREATE_, or PROCESS_SPAWN_, and correct the error.

```
22    BACKUP CREATION SPECIFIED, BUT CALLER IS UNNAMED
```

Cause. The operating system tried to create a backup process, but the caller was unnamed.

Effect. No process is created.

Recovery. Run the caller as a named process, or rebind its program file as “must run named” by using the Binder.

```
24    DEFINE CONTEXT PROPAGATION ERROR
```

Cause. An error occurred when existing DEFINEs were being propagated. The error-detail information contains either a file-system error, a DEFINE error number, or the error subcode 2, which indicates that an excessive number of DEFINEs were to be propagated. File-system errors are described in Section 2, File-System Errors. DEFINE errors are described in Section 4, DEFINE Errors.

Effect. No process is created.

Recovery. Retry the call. If errors recur, contact your service provider.

```
26    DYNAMIC IOP ERROR
```

Cause. This error is returned only to privileged callers, or unprivileged callers attempting to use certain privileged features. On G series, it can be returned by an attempt to create a D-series I/O Process.

Effect. No process is created.

Recovery. Run software that is properly configured for your system.

```
27    ILLEGAL PFS SIZE
```

Cause. The size specified for the process file segment (PFS) is invalid.

Effect. No process is created.

Recovery. Specify a valid PFS size either in the RUN command or with the Binder or `nld` utility. Then try again. (This error is not generated as of G06.12.)

```
29    UNABLE TO ALLOCATE A PRIV STACK FOR THE PROCESS
```

Cause. There was no segment available for the priv stack of a TNS/R native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
30    UNABLE TO LOCK THE PRIV STACK FOR THE PROCESS
```

Cause. There was not enough physical memory free to lock the priv stack for a TNS/R native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
31    UNABLE TO ALLOCATE A MAIN STACK FOR THE PROCESS
```

Cause. There was no segment available for the main stack of a TNS/R native process.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
32    UNABLE TO LOCK THE MAIN STACK OF A NATIVE I/O PROCESS
      THAT IS CREATED DYNAMICALLY
```

Cause. There was not enough physical memory free to lock the main stack for a TNS/R native process. This error is only returned to some privileged callers.

Effect. No process is created.

Recovery. Try a different processor or wait until the system load decreases and then retry the existing call.

```
33    SECURITY INHERITANCE FAILURE
```

Cause. An error occurred during an attempt to obtain or propagate security identity information. The error-detail information contains the number of the file-system error that occurred.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information. If the file-system error indicates an invalid operation or an invalid parameter, the problem might be caused by a version mismatch between the NonStop operating system and Standard Security. Correct your system configuration if this is the case. Otherwise, the problem is likely to be an internal error that should be reported to your service provider.

```
35    INTERNAL PROCESS CREATION ERROR
```

Cause. An internal error occurred in the PROCESS_LAUNCH_, PROCESS_CREATE_, or PROCESS_SPAWN_ procedure.

Effect. No process is created.

Recovery. Make a note of the error subcode returned in the error-detail information and retry the call. Contact your service provider.

```
36    CHILD'S PFS ERROR
```

Cause. An error occurred in the specification of the process file segment (PFS). The error-detail information contains the error number of the file-system error that occurred.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information. For example, if file-system error 31, “Unable to obtain file-system buffer space,” is returned, specify a larger PFS size for the process.

```
37    UNABLE TO ALLOCATE GLOBAL DATA FOR THE PROCESS
```

Cause. Process memory-segment limit exceeded.The system was unable to allocate the global data segment of a native process. If non-zero, *error-detail* indicates:

1. Insufficient swap space available from KMSF
2. Overlap of Vaddrs
3. Too many segments loaded

Effect. No process is created. The name of the file for which the error occurred may have been reported to the home terminal.

Recovery. Contact your local service provider.

```
38    UNABLE TO LOCK IOP GLOBAL DATA FOR THE PROCESS
```

Cause. The system was unable to lock the global data segments of a native I/O process. This error is only returned to some privileged callers.

Effect. No process is created.

Recovery. Stop processes or wait until the system load decreases before attempting to run this process.

```
40    THE MAIN STACK MAXIMUM VALUE, SPECIFIED EITHER BY THE
      PROCEDURE CALL OR BY THE OBJECT FILE, IS TOO LARGE
```

Cause. The main stack maximum value is too large.

Effect. No process is created.

Recovery. Either change the procedure call (if applicable) or use the `nld` utility to change the object file to specify a new value that is within the valid range.

```
41    THE HEAP MAXIMUM VALUE, SPECIFIED EITHER BY THE
      PROCEDURE CALL OR BY THE OBJECT FILE, IS TOO LARGE
```

Cause. The heap maximum value is too large.

Effect. No process is created.

Recovery. Either change the procedure call (if applicable) or use the `nld` utility to change the object file to specify a new value that is within the valid range.

```
42    THE SPACE GUARANTEE VALUE, SPECIFIED EITHER BY THE
      PROCEDURE CALL OR BY THE OBJECT FILE, IS TOO LARGE
```

Cause. The space guarantee value is too large.

Effect. No process is created.

Recovery. Either change the procedure call (if applicable) or use the `nld` utility to change the object file to specify a new value that is within the valid range.

```
43    THE PROCESS CREATION REQUEST SPECIFIES TWO FILES THAT
      CONTAIN THE SAME SHARED RUN-TIME LIBRARY (SRL) NAMES
```

Cause. The process creation request specifies duplicate shared run-time libraries (SRLs); the error-detail information contains the numbers of the duplicate SRLs in the form *xxyy* (where *xx* is the first SRL and *yy* is the duplicate SRL).

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file.

```
44    UNABLE TO FIND A SHARED RUN-TIME LIBRARY (SRL)
SPECIFIED
      BY THE PROGRAM FILE
```

Cause. The system was unable to find a shared run-time library (SRL) specified by the program file; the error-detail information contains the SRL number of the SRL that could not be found.

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
45    UNABLE TO FIND A SHARED RUN-TIME LIBRARY (SRL)
SPECIFIED
      BY ANOTHER SRL
```

Cause. The system was unable to find a shared run-time library (SRL) specified by another SRL; the error-detail information contains the SRL numbers of the two SRLs in the form *xxyy* (where *xx* is the SRL that specifies the *yy* SRL).

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
46    THE PROCESS CREATION REQUEST SPECIFIES TOO MANY SHARED
      RUN-TIME LIBRARIES (SRLS)
```

Cause. The process creation request specifies too many shared run-time libraries (SRLs); the error-detail information contains the maximum number of SRLs that can be specified.

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file, using no more than 32 SRLs, the maximum number of SRLs permitted.

```
47    THE PROGRAM FILE REQUIRES FIXUPS TO A SHARED RUN-TIME
      LIBRARY (SRL) BUT THE PROGRAM FILE IS CURRENTLY RUNNING
```

Cause. The program file requires fixups to a shared run-time library (SRL) that is unavailable because it is running; the error-detail information contains the SRL number of the running SRL.

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
48    A SHARED RUN-TIME LIBRARY (SRL) REQUIRES FIXUPS TO
      ANOTHER SRL
```

Cause. A shared run-time library (SRL) requires fixups to another SRL that is unavailable because it is running; the error-detail information contains the SRL numbers of the two SRLs in the form *xxyy* (where *xx* is the SRL that requires the fixup to the running *yy* SRL).

Effect. No process is created.

Recovery. Use the original object files and the `nld` utility to re-create the program file. If the error recurs, contact your service provider.

```
49    SECURITY VIOLATION. THE PROGRAM FILE IS NOT LICENSED
BUT
      A SHARED RUN-TIME LIBRARY (SRL) CONTAINING INSTANCE
DATA
      IS LICENSED
```

Cause. A security violation occurred. The program file is not licensed but a shared run-time library (SRL) is licensed and has instance data; the error-detail information contains the SRL number of the licensed SRL.

Effect. No process is created.

Recovery. License the program file, if possible.

```
50    SECURITY VIOLATION.  EITHER THE PROGRAM FILE OR SHARED
      RUN-TIME LIBRARY (SRL) IS LICENSED BUT A SHARED RUN-
TIME
      LIBRARY (SRL) IS NOT LICENSED
```

Cause. A security violation occurred. Either the program file or a shared run-time library (SRL) is licensed, but another SRL is not licensed; the error-detail information contains the SRL number of the unlicensed SRL.

Effect. No process is created.

Recovery. Either remove the licensing of the program, if possible, or use the `nld` utility to specify another SRL.

```
51    THE PROGRAM FILE REQUIRES A SYMBOL FROM A SHARED RUN-
      TIME LIBRARY (SRL) BUT THE SRL IS NOT EXPORTING IT
```

Cause. The program file requires a symbol from a shared run-time library (SRL) but the SRL is not exporting it; the error-detail information contains the SRL number of the SRL that does not export the required symbol.

Effect. No process is created.

Recovery. Use the `nld` utility to specify another SRL that contains the desired data block.

```
52    THE SPECIFIED VERSION, Z^VERSION, OF THE
      ZSYS^DDL^PLAUNCH^PARMS STRUCTURE IS NOT SUPPORTED
```

Cause. (This error is returned only by PROCESS_LAUNCH_ .) The version Z^VERSION of the ZSYS^DDL^PLAUNCH^PARMS structure is not supported.

Effect. No process is created.

Recovery. Recode or recompile and rebuild the program.

```
53    THE SPECIFIED VERSION, Z^VERSION, OF THE
      ZSYS^DDL^PLAUNCH^PARMS STRUCTURE IS INCOMPATIBLE WITH
      THE SPECIFIED LENGTH,Z^LENGTH,OF THE STRUCTURE
```

Cause. (This error is returned only by PROCESS_LAUNCH_ .) The version Z^VERSION of the ZSYS^DDL^PLAUNCH^PARMS structure is incompatible.

Effect. No process is created.

Recovery. Recode or recompile and rebuild the program.

```
54    INTERNAL PROCESS CREATION ERROR
```

Cause. A privileged interface to the PROCESS_LAUNCH_ procedure is invalid.

Effect. No process is created.

Recovery. Recode or recompile and rebuild the program.

```
55    THE SPECIFIED SPACE GUARANTEE CANNOT BE ALLOCATED
```

Cause. The space guarantee specified in Z^SPACE^GUARANTEE (PROCESS_LAUNCH_) or Z^SPACEGUARANTEE (PROCESS_SPAWN_) cannot be allocated.

Effect. No process is created.

Recovery. Either use the `nld` utility to shrink the guaranteed swap space specification, use NSKCOM to add additional swap space for the target processor, or run on another processor.

```
56    INTERNAL PROCESS CREATION ERROR
```

Cause. An internal error occurred.

Effect. No process is created.

Recovery. Retry the call. If this error recurs, contact your service provider.

```
57    A SHARED RUN-TIME LIBRARY (SRL) HAS UNDEFINED EXTERNALS
```

Cause. A shared run-time library (SRL) has undefined externals; the error-detail information contains the SRL number of the SRL that has undefined externals.

Effect. No process is created.

Recovery. Check that the SRL versions match and obtain matching versions if they don’t; otherwise, contact your service provider.

```
58    INTERNAL PROCESS CREATION ERROR
```

Cause. An internal error occurred.

Effect. No process is created.

Recovery. Retry the call. If this error recurs, contact your service provider.

```
59    INTERNAL PROCESS CREATION ERROR
```

Cause. An internal error occurred.

Effect. No process is created.

Recovery. Retry the call. If this error recurs, contact your service provider.

```
60    SECURITY VIOLATION; A SHARED RUN-TIME LIBRARY (SRL)
      CONTAINING CALLABLE PROCEDURES MUST BE LICENSED TO BE
      USED BY CALLABLE OR PRIVILEGED CODE
```

Cause. A security violation occurred; a shared run-time library (SRL) containing callable procedures must be licensed to be used by callable or privileged code.

Effect. No process is created.

Recovery. Rebuild the program file. If the error recurs, contact your service provider.

```
61   UNABLE TO ALLOCATE MEMORY FROM SYSTEM POOL
```

Cause. Memory for a needed buffer was unavailable in the system pool in the processor.

Effect. No process is created.

Recovery. Try again or select another processor. If the problem persists, run the PEEK program in the processor where process creation failed and determine new memory needs.

```
62   SYMBOLIC REFERENCE TARGET/SOURCE TYPE MISMATCH
```

Cause. An attempt was made to resolve a procedure address to a data area or a data address to a procedure area.

Effect. No process is created, and a message is printed on the home terminal. For example:

```
PID: \SAT.1,284 \SAT.$DATA.SRLVOL.NULCLIE2 (ELF)
Native UL Symbolic Reference Error: Target/Source Type Mismatch:
Prg: \SAT.$DATA.SRLVOL.NULCLIE2 -> X_INT (DATA)
*ERROR* PROCESS_CREATE_ Error: 62
```

Recovery. Use the original object files and the `nld` utility to recreate the program file.

```
63   EXTERNAL DATA REFERENCES NOT RESOLVED TO ANY USER/SYSTEM
     LIBRARY
```

Cause. An anonymous data symbol reference was not located in a UL, Native UL, or system library.

Effect. No process is created, and a message is printed on the home terminal. For example:

```
PID: \SAT.1,284 \SAT.$DATA.SRLVOL.NULCLIE2 (ELF)
External References Not Resolved to Any User/System Library:
Prg: \SAT.$DATA.SRLVOL.NULCLIE2 -> X_INT (DATA)
*ERROR* PROCESS_CREATE_ Error: 63
```

Recovery. Either correct the coding error, use `nld` utility to add the data to the data area or user library, or refer the call to the data that is already present in the application program.

```
64   UNABLE TO HONOR FLOATTYPE ATTRIBUTE
```

Cause. The system couldn't honor the floattype attribute requested by a process-creation request. The error-detail information contains an error subcode that indicates the nature of the error. These error subcodes are listed in Table 6-3.

Effect. No process is created.

Recovery. Take corrective action as indicated by the subcode. As of G06.20 this code is reported only for a program file; earlier releases use it also for libraries. These error subcodes are listed in Table 6-3.

Table 6-3. Error Subcodes for Process Creation Error 64

Subcode	Meaning
1	IEEE floating-point not supported by processor. Run your program on a processor model that supports IEEE floating-point operation. If the process does not operate on floating-point data, but was mistakenly marked as using IEEE floating-point, use the linker to force the floattype attribute to tandem_float. The floattype attribute can be specified to the linker using the -set command at link time, or the -change command to modify in an existing file. The linker is the nld utility for a non-PIC program, or the ld utility for a PIC program.
2	Unrecognized floating-point specification in file. The program or library file specified to create the process contains incorrect or corrupt information about the floattype. Recompile the object file(s) using a valid floattype.
3	Floattype attribute of user library conflicts with that of the program (pre-G06.20 only). Recompile and relink the object file(s) using the same floattype attribute for all, or if the library does not operate on floating-point data, force its floattype attribute to neutral_float. The floattype attribute can be specified to the nld utility using the -set command at link time, or the -change command to modify in an existing file.

```
65   REFERENCED SRL IS ALREADY IN USE
```

Cause. A shared run-time library (SRL) requires a symbol from another SRL but cannot be referenced because that SRL is already in use.

Effect. No process is created.

Recovery. Use the nld utility to specify the another SRL that contains the correct symbol.

```
66   UNABLE TO HONOR FLOATTYPE ATTRIBUTE OF A USER LIBRARY
```

Cause. The system couldn't honor the floattype attribute for the user library requested by a process-creation request. The error-detail information contains an error subcode that indicates the nature of the error. The error-detail indicates:
2: Unrecognized floating-point specification in file.
4: This library specified Tandem floating-point, which mismatches the program.
5: This library specified IEEE floating-point, which mismatches the program.

Effect. No process is created. If the user library is PIC, its name will have been reported to the home terminal.

Recovery. Take corrective action as indicated by the subcode.
For subcode 2: The library file specified to create the process contains incorrect or

corrupt information about the floattype. Recompile the object file(s) using a valid floattype.

For subcode 4: Recompile and relink the object file(s) using the same floating-point type for all, or if the library does not operate on floating-point data, force its floattype attribute to neutral_float. The floattype attribute can be specified to the linker using the -set command at link time, or the -change command to modify in an existing file. The linker is the nld utility for a non-PIC user library (SRL), or the ld utility for a PIC user library (DLL). Note that the library specified Tandem floating-point.

For subcode 5: Recompile and relink the object file(s) using the same floating-point type for all, or if the library does not operate on floating-point data, force its floattype attribute to neutral_float. The floattype attribute can be specified to the linker using the -set command at link time, or the -change command to modify in an existing file. The linker is the nld utility for a non-PIC user library (SRL), or the ld utility for a PIC user library (DLL). Note that the library specified IEEE floating-point.

```
67   UNABLE TO HONOR FLOATTYPE ATTRIBUTE OF A DLL
```

Cause. The system couldn't honor the floattype attribute for a DLL. The error-detail information contains an error subcode that indicates the nature of the error. The error-detail indicates:

2: Unrecognized floating-point specification in file.
4: This library specified Tandem floating-point, which mismatches the program.
5: This library specified IEEE floating-point, which mismatches the program.

Effect. No process is created. The name of the DLL for which the error occurred will have been reported to the home terminal.

Recovery. Take corrective action as indicated by the subcode.

For subcode 2: The DLL file specified to create the process contains incorrect or corrupt information about the floattype. Recompile the object file(s) using a valid floattype.

For subcode 4: Recompile and relink the object file(s) using the same floating-point type for all, or if the DLL does not operate on floating-point data, force its floattype attribute to neutral_float. The floattype attribute can be specified to the ld utility; use the -set command at link time, or the -change command to modify in an existing file. Note that the DLL specified Tandem floating-point.

For subcode 5: Recompile and relink the object file(s) using the same floating-point type for all, or if the DLL does not operate on floating-point data, force its floattype attribute to neutral_float. The floattype attribute can be specified to the ld utility; use the -set command at link time, or the -change command to modify in an existing file. Note that the DLL specified IEEE floating-point.

```
68   A DEFINE = _RLD IS PRESENT BUT NOT VALID
```

Cause. *error-detail* indicates:
0 - The define is not of class SEARCH.
2055 - An attribute other than CLASS or SUBVOL0 is specified.
Otherwise - as reported by the DEFINEINFO function.

Effect. No process is created.

Recovery. Take action as indicated by the error subcode.

```
69   A FILE SYSTEM ERROR WAS ENCOUNTERED ON THE RUNTIME
LOADER LIBRARY (rld)
```

Cause. A file system error occurred on the runtime loader library while attempting the process creation. The error-detail contains a file-system error number.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
70   INVALID FILE FORMAT IN THE RUNTIME LOADER LIBRARY (rld)
```

Cause. An error was detected in the file format of the runtime loader library (RLD). The error-detail information contains an error subcode that indicates the invalid file format error. These errors are listed in Table 6-2.

Effect. No process is created.

Recovery. Take corrective action as indicated by the subcode. In many cases, the object file was built improperly or corrupted; it should be relinked and perhaps recompiled.

```
71   ERROR LOADING OR RUNNING THE RUNTIME LOADER (rld)
```

Cause. A failure occurred while loading or running the runtime loader (RLD). Error details other than the following indicate that rld was not constructed or installed correctly.
9- The process abended while rld was running.
10- The processed stopped while rld was running.
11- RLD is licensed.
12- rld returned an out-of-range error value to the operating system.
16- The export digest of the file does not match the export digest of the impImp file in memory.
19- The user attempted to use RLD as a user library. This is not supported.
22- RLD began processing, but did not complete the update of a loadfile.

Effect. No process is created.

Recovery. For subcode 11, unlicense RLD and reload the system. RLD should not require being licensed. For subcode 16, verify that the copy of RLD you used is the correct one. The set of public DLLs, including ZRLDDLL, should be installed and preset for use with the implicit DLLs in this SYS*nn*. There is no recovery for subcode error 19. For other subcodes, contact your service provider, reporting the error code and the error detail.

```
72   THE RUNTIME LOADER (rld) REPORTED AN INTERNAL OR
UNRECOGNIZED ERROR.
```

Cause. The runtime loader (RLD) reported an internal or unrecognized error.

Effect. No process is created.

Recovery. Contact your service provider and report the error and also the error detail.

```
74   UNRESOLVED REFERENCE TO A FUNCTION
```

Cause. There was an unresolved reference to a function, so the process was not created. Contrast with error 14, which is a warning. (For DLLs and their client programs, unresolved function references are disallowed by default, but other options can be specified at link or run time.)

Effect. No process is created. Information such as the name of the unresolved function and the caller of it may be reported to the home terminal.

Recovery. Use the information reported on the home terminal, and attempt to fix the problem. (Either fix the code in the caller or the callee, or get a version of code that does not make the faulty call.)

```
75   A FILE SYSTEM ERROR WAS ENCOUNTERED ON A DLL
```

Cause. A file system error occurred on a DLL while attempting the process creation. The error-detail contains a file-system error number.

Effect. No process is created. The name of the DLL for which the error occurred will have been reported to the home terminal.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information.

```
76   INVALID FILE FORMAT IN A DLL
```

Cause. An error was detected in the file format of a DLL. The error-detail information contains an error subcode that indicates the invalid file format error. These errors are listed in Table 6-2.

Effect. No process is created. The name of the DLL for which the error occurred will have been reported to the home terminal.

Recovery. Take corrective action as indicated by the subcode. In many cases, the object file was built improperly or corrupted; it should be relinked and perhaps recompiled.

```
 77   AN OBJECT FILE COULD NOT BE LOADED
```

Cause. *error-detail* contains the details:
 1 - A DLL requires a PIN < 255 in a process with a higher PIN.
 2 - A licensed or privileged object file does not use localized import.
 3 - A public library requires fixed address space that is unavailable in this process.
 4 - Insufficient address range is available to load the file.
 5 - The loader lacks authority to load a licensed object file.
 6 - The process has exceeded the maximum number of memory segments.
 7 - A Globalized symbol required by a PIC program or DLL is defined in more than one public library. (Globalized symbols are those identified by the native C++ compiler as possibly having multiple definitions, of which the system must select one to be used by all object modules loaded.)
 8 - The C++ version of the specified library conflicts with one or more loadfiles loaded for this process. Mixed C++ dialect versions are not allowed.
 9- The loadfile is not licensed; license is required.
10- A DLL for this process is licensed or privileged and has unprotected data which requires that all loadfiles in the process be licensed. There is at least one unlicensed loadfile in the process.
11- A licensed DLL or a privileged program refers to an unlicensed DLL.
12- A process with a licensed but unprivileged program attempted to load an unlicensed non-public DLL.
13- A licensed or privileged loadfile has globalized symbols.
14- The loadfile was specified as dataResident and is not licensed, has no callable functions, and is not a program with a priv entry point.
15- This process can only be run by the local super ID.
16- RLD failed to pass to the operating system a function pointer necessary to process the initialization functions, constructor callers, destructor callers, or termination functions specified to the linker.
17- The specified loadfile was built with linker option -no_runtime_fixup, but it is not preset to load with the symbol bindings available on this system or in this process.
18- The loadfile was built to use an Application Binary Interface version that is not supported.

Effect. No process is created. The name of the file involved may be reported on the home terminal.

Recovery. This depends on the error detail.
 1- Specify 'highpin off' when running the Program (or rebuild the DLL to not need a low pin)
 2- If this file must be licensed, you must rebuild it specifying localized import rather than globalized or semi-globalized

3 - Probably indicates a problem in the way the library was built. Contact your service provider. Report the file name, error code, and the error detail.
4 - Probably indicates that either too many segments or too many large segments are already loaded into the process. To get around this, you may have to try to run the program with fewer libraries.
5 - If this file must be licensed, it cannot be loaded without “RLD authority”; you cannot open it dynamically using dlopen().
6 - Probably indicates that either too many segments or too many large segments are already loaded into the process. To get around this, you may have to try to run the program with fewer libraries.
7 - If an inadvertent collision of symbol names occurred, the PIC source must be changed, recompiled, and relinked using a unique symbol. Provide this information to the supplier of the PIC program or DLL. If the same symbol is exported by two public libraries, and it is proper for both of them to be loaded into the same process, report the problem to your service provider. Avoid loading redundant or unnecessary public SRLs; for example, ensure that all files loaded in the process depend upon the same C++ runtime and/or tools.h++++ libraries.
8 - Don't load both a Version 3 C++ loadfile and any Version 1 or Version 2 C++ loadfile into the same process. If possible, relink a file that used a conflicting C++ version file so that it uses the same C++ version as other loadfiles loaded into the process.
9-You must license this loadfile to continue using it in this manner.
10- Avoid this situation either by avoiding unprotected data or by licensing the other loadfiles in the process when feasible and appropriate. Note that libraries with unprotected data cannot be licensed and then used with an unlicensed program. A loadfile’s data will be protected if it has no variable data, or if it is constructed using the -instance_data data2hidden command to the linker. The former makes the library’s global variables read-only at user privilege; the latter makes them invisible at user privilege. Therefore, the variables are fully accessible only from privileged code. Note that licensed DLLs must have localized import and cannot have globalized symbols.
11- License the referenced DLL when possible. Note that libraries with unprotected data cannot be licensed and then used with an unlicensed program. This does not apply to licensing most public DLLs such as the language-support libraries.
12- License the DLL when possible. Note that libraries with unprotected data cannot be licensed and then used with an unlicensed program. This does not apply to licensing most public DLLs such as the language-support libraries.
13- To license this loadfile, it cannot contain global symbols.
14- If the loadfile must be dataResident, license it. Otherwise relink it without specifying that it be dataResident.
15- The process must be created by a person or process using the locally authenticated super ID.
16- Contact your service provider. Report the error code and the error detail.
17-Relink the file if the environment changes, or don't set the option if the same file must be used simultaneously in different environments. Contact your service provider. Report the file name, error code, and the error detail.
18- Recompile the source code with current compilers and relink the loadfile with the

current linker. Contact your service provider. Report the file name, error code, and the error detail.

```
78   AN UNSUPPORTED OPERATION WAS ATTEMPTED
```

Cause. *error-detail* contains the details:
1 - A PIC program attempted to load an SRL other than a public SRL.
2 - A PIC program or DLL was licensed.
3 - A User Library supplied for a PIC program was not a DLL.
4 - A public SRL requires another library that is not a public SRL.
5 - A public SRL is not a hybrid DLL-SRL.
The *error-detail* values 4 and 5 imply an incorrect installation of the public SRLs.
6 - The specified library uses Version 1 C++, which is not supported with a PIC Program.
7- RLD cannot be on the liblist of any file loaded when a) the program has a priv entry point, or b) the program file is licensed.

Effect. No process is created. The name of the file involved may be reported on the home terminal.

Recovery. This depends on the error detail.
1 - Only attempt to use public SRLs and DLLs when running a PIC program. The home terminal may indicate both the file that attempted to use the invalid file and the invalid file. Replace one of these files as needed to follow the requirements.
2 - Licensed PIC Programs and DLLs are not supported on this system. The home terminal may indicate which file was used. If it did not need to be licensed, unlicense it, and try again.
3 - Only a DLL file can be used as a user library with a PIC program. If possible, replace the user library with a valid DLL file, and try again.
4 and 5 - An incorrect installation of the public SRLs was done. Contact your service provider.
6 - Do not include any loadfile that requires a Version 1 C++ loadfile when loading a PIC Program. If possible, relink the loadfile that used the Version 1 loadfile so that it uses a Version 2 or Version 3 C++ loadfile in place of the Version 1 C++ loadfile.
7- Fix the application so that RLD is not included on a liblist. Such a process cannot call dlopen().

```
79   A RESOURCE LIMITATION WAS DETECTED BY THE RUNTIME LOADER
(rld)
```

Cause. *error-detail* contains the details:
1 - The RLD heap attempted to exceed available KMSF space.
2 - The RLD heap attempted to exceed its allocated address range.
3 - RLD limit of handles was exhausted. This error is reported for dynamic loading, not at process creation.
4 - The process limit for keys is exhausted. Keys are an operating-system resource used by the loader.

Effect. No process is created.

Recovery. For detail 3:
Each call on dlopen() generates a new handle; avoid a large number of calls specifying the same path argument.
For the others:
Contact your service provider, and report the error and the error detail.

```
80   ERROR LOADING A DROP-IN PROGRAM
```

Cause. A failure occurred while loading or running a program that must be “dropped in” rather than run through RLD. Error details indicate that the drop-in program is not constructed or installed correctly.
16- The export digest of the file does not match the export digest of the impImp file in memory.

Effect. No process is created.

Recovery. For subcode 16 verify that the right copy of the drop-in program is being used. It must have been preset for use with the implicit DLLs in the SYS*nn*. For other subcodes, contact your service provider, reporting the error code and the error detail.

```
81   ERROR LOADING OR RUNNING THE TNS EMULATOR
```

Cause. A failure occurred while loading or running the TNS Emulator. Error details other than the following indicate that the TNS Emulator is not constructed or installed correctly.
16- The export digest of the file does not match the export digest of the impImp file in memory.

Effect. No process is created.

Recovery. For subcode 16 verify that the right copy of the TNS emulator is being used. The set of public DLLs, including ZTNSDLL, should be installed and preset for use with the implicit DLLs in this SYS*nn*.For other subcodes, contact your service provider, reporting the error code and the error detail.

```
82   A DEFINE=_TNS IS PRESENT BUT NOT VALID
```

Cause. A DEFINE is recognized by the systems, but it is not a valid DEFINE. Error details are:
0 - The define is not class SEARCH
2055 - An attribute other than CLASS or SUBVOL0 is specified
All other details are as reported by the DEFININFO function.

Effect. No process is created.

Recovery. Take corrective action as indicated by the error subcode.

```
83   A FILE SYSTEM ERROR WAS ENCOUNTERED ON THE TNS EMULATOR
```

Cause. A file system error occurred on the TNS Emulator while attempting process creation. The error detail contains a file system error number.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors for further details about the file-system error returned in the error detail information.

```
84   INVALID FILE FORMAT IN THE TNS EMULATOR
```

Cause. An error is detected in the file format of the TNS Emulator. The error detail information contains an error subcode that indicates the invalid file format. These errors are listed in Table 6-2 on page 6-7.

Effect. No process is created.

Recovery. Take corrective action as indicated by the error subcode. In many cases, the object file was improperly built or corrupted. Consider relinking and recompiling the file.

```
99   ERROR IN PRELOADING A PUBLIC DLL
```

Cause. A failure occurred while attempting to preload a public DLL specified in the zreg file. Error details are:
1- The export digest of the public DLL is not a match to the export digest found in the specified zreg file.
2- The license value of the public DLL is not a match to the license value found in the specified zreg file.
3- The public DLL is licensed and has unprotected data.
4- The public DLL is not preset.
5- The public DLL has a priv or callable Main procedure.
6- The public DLL does not support highpin.
7- The public DLL is not owned by super ID.
8- The public DLL has callable procedures and is not licensed.
9- A public DLL with this name has already been preloaded (duplicate name in zreg).
10- The text, data, or gateway of the public DLL overlaps that of another public DLL.
11- The export digest attribute for this public DLL is missing from the zreg file.

Effect. The public DLL is not preloaded into the public library table.

Recovery. Contact your service provider reporting the error code and error detail if you are using a zreg file and public DLL set that was created by PLINSTL and has not been modified. If you have been attempting to adjust the public DLL set manually (not recommended), fix or replace the loadfile based on the problem reported in the error detail, or run PLINSTL to create an appropriate set of public DLLs.

```
104   UNABLE TO CREATE OSS PROCESS
```

Cause. (This error is returned only by PROCESS_SPAWN_.) An OSS process cannot be created because there are insufficient resources.

Effect. No process is created.

Recovery. The recovery method depends on the error-detail code received:

Recovery. 1: Request TNS floating point in your creation request, or upgrade to a release and processors that support IEEE floating point.

Recovery. 2: Change your process create request to one of the supported floating point types.

Recovery. 3: Recompile the object file(s) with the same floating point type for all.

```
106   OSS START INTERPRETER ERROR
```

Cause. (This error is returned only by PROCESS_SPAWN_.) The *oss-program-file* parameter is an interpreter shell script that cannot be started. The error-detail information contains the error number of the file-system error that occurred.

Effect. No interpreter script file is started.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the error returned in the error-detail information.

```
107   OSS STATIC VARIABLE ERROR
```

Cause. (This error is returned only by PROCESS_SPAWN_.) An error occurred during the allocation of user data space for static variables used by the system library. The error-detail information contains the error number of the file-system error that occurred.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the error returned in the error-detail information.

```
108   THE CALLING PROCESS IS NOT OSS
```

Cause. A Guardian process attempted to call an OSS-only process-creation function: fork, exec..., tdm_fork, tdm_exec..., tdm_spawn...

Effect. No process is created.

Recovery. Make the call from an OSS process.

```
110   OSS CURRENT WORKING DIRECTORY ERROR
```

Cause. (This error is returned only by PROCESS_SPAWN_.) The current working directory for the new process could not be obtained. The current working directory is specified in the ZSYS-DDL-FDINFO.Z-CWD field of the *`fdinfo`* parameter. If it is null, then the caller's current working directory is used. If the caller does not have a current working directory, then the caller's default volume is used. The error-detail information contains either an OSS `errno` value or the error number of the file-system error that occurred.

Effect. No process is created.

Recovery. If the error-detail information contains the OSS `errno` value ENOENT (4002), then the specified current working directory does not exist or does not exist on the current system. Make sure that the default volume exists and is local to the calling process. For other error values, refer to Section 2, File-System Errors, for corrective action for the error returned in the error-detail information.

```
111   OSS FILE DESCRIPTOR DUP ERROR
```

Cause. (This error is returned only by PROCESS_SPAWN_.) One of the file descriptors specified to be duplicated with the OSS `dup()` function that was passed in the *`fdinfo`* parameter could not be duplicated. The ZSYS-DDL-PROCESSRESULTS.Z-TPCDETAIL field of the *`process-results`* parameter contains the index into the ZSYS-DDL-FDINFO.Z-FDENTRY structure to identify which of the file descriptors failed to be duplicated. The ZSYS-DDL-PROCESSRESULTS.Z-ERRNO field of the *`process-results`* parameter contains an OSS `dup()` function `errno` value.

Effect. No process is created.

Recovery. Determine corrective action by referring to the *Open System Services System Calls Reference Manual* for a description of the OSS `dup()` function `errno` values.

```
112   OSS FILE DESCRIPTOR OPEN ERROR
```

Cause. (This error is returned only by PROCESS_SPAWN_.) One of the file descriptors specified to be opened with the OSS `open()` function that was passed in the *`fdinfo`* parameter could not be opened. The ZSYS-DDL-PROCESSRESULTS.Z-TPCDETAIL field of the *`process-results`* parameter contains the index into the ZSYS-DDL-FDINFO.Z-FDENTRY structure to identify which of the file descriptors failed to be opened. The ZSYS-DDL-PROCESSRESULTS.Z-ERRNO field of the *`process-results`* parameter contains an OSS `open()` function `errno` value.

Effect. No process is created.

Recovery. Determine corrective action by referring to the *Open System Services System Calls Reference Manual* for a description of the OSS `open()` function `errno` values.

```
113   OSS FILE DESCRIPTOR OPEN TIMEOUT ERROR
```

Cause. (This error is returned only by PROCESS_SPAWN_.) The timeout value in the ZSYS-DDL-FDINFO.Z-TIMEOUT field of the *fdinfo* parameter was reached before the file descriptors specified in the *fdinfo* parameter could be opened.

Effect. No process is created.

Recovery. Recovery, if any, is application dependent. Recovery actions include but are not limited to the following: select a longer timeout value, select a timeout value of 0 to allow the PROCESS_SPAWN_ call to return to the application without waiting for files to be opened or duplicated, select a different file to be opened or duplicated, or select a different open flag such as OSSOPEN^NONBLOCK to open the file for nonblocked access.

```
114   CANNOT CREATE OSS PRIV PROCESS
```

Cause. (This error is returned only by PROCESS_SPAWN_.) The process cannot be created because privileged OSS processes are not supported.

Effect. No process is created.

Recovery. Specify a program file that is not to be run as a privileged process when calling the PROCESS_SPAWN_ procedure.

```
115   UNABLE TO ALLOCATE GLOBAL DATA OR HEAP FOR THE PROCESS
      (PROCESS_SPAWN_ ONLY)
```

Cause. (This error is returned only by PROCESS_SPAWN_ .) The system was unable to allocate global data or heap for the process.

Effect. No process is created.

Recovery. Run on another processor or try later.

```
116   UNABLE TO PROPAGATE SHARED RUN-TIME LIBRARY (SRL) DATA
      (PROCESS_SPAWN_ ONLY).
```

Cause. (This error is returned only by PROCESS_SPAWN_ .) The system was unable to propagate shared run-time library (SRL) data.

Effect. No process is created.

Recovery. Run on another processor or try later.

```
3xx   INVALID FILE FORMAT ON SHARED RUN-TIME LIBRARY (SRL)
      NUMBER XX
```

Cause. An error was detected in the file format of shared run-time library(SRL) number xx. The error-detail information contains an error subcode that indicates the invalid file format error. See Table 6-2 on page 6-7 for possible values.

Effect. No process is created.

Recovery. Take corrective action as indicated by the subcode. In many cases, the object file was built improperly or corrupted; it should be relinked and perhaps recompiled.

```
5xx   FILE-SYSTEM ERROR ON SHARED RUN-TIME LIBRARY (SRL)
      NUMBER XX
```

Cause. There was a file-system error on shared run-time library (SRL) number *xx*; error-detail information contains a file-system error number.

Effect. No process is created.

Recovery. Refer to Section 2, File-System Errors, for corrective action for the file-system error returned in the error-detail information. If the problem persists, contact your service provider.

```
1100-1499  INTERNAL ERROR WITHIN A MODULE OF THE OPERATING
SYSTEM
```

Cause. An internal error was detected within a module of the operating system.

Effect. No process is created.

Recovery. Report the internal error and any error detail that accompanied it to your service provider.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive an error list in a response. HP subsystems return such an error list when, in performing your request, they call the PROCESS_LAUNCH_ or PROCESS_CREATE_ procedure directly or indirectly and an error occurs on the call.

The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every PROCESS_LAUNCH_ and PROCESS_CREATE_ error list, identifies the procedure that was called. Its value is ZGRD-VAL-PROCESS-CREATE (22) for PROCESS_CREATE_ and ZGRD-VAL-PROCESS-LAUNCH (25) for PROCESS_LAUNCH_.

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a PROCESS_LAUNCH_ or PROCESS_CREATE_ error list:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in the error-list description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about tokens and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

22: ZGRD-VAL-PROCESS-CREATE

A call to PROCESS_CREATE_ failed due to an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST              token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR             token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR          token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST              token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-XOBJECTFILE       token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XLIBRARYFILE      token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XSWAPFILE         token-type ZSPI-TYP-STRING.
   ZGRD-TKN-EXTSWAPFILE       token-type ZSPI-TYP-STRING.
   ZGRD-TKN-PRIORITY          token-type ZSPI-TYP-UINT.
   ZGRD-TKN-XCPU              token-type ZSPI-TYP-UINT.
   ZGRD-TKN-PROCESSHANDLE     token-type ZSPI-TYP-PHANDLE.
   ZGRD-TKN-ERRORDETAIL       token-type ZSPI-TYP-INT.
   ZGRD-TKN-NAMEOPTION        token-type ZSPI-TYP-ENUM.
   ZGRD-TKN-XPROCESSNAME      token-type ZSPI-TYP-STRING.
   ZGRD-TKN-PROCESSDESCR      token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XHOMETERM         token-type ZSPI-TYP-STRING.
   ZGRD-TKN-JOBID             token-type ZSPI-TYP-INT.
   ZGRD-TKN-XMEMORYPAGES      token-type ZSPI-TYP-INT.
   ZGRD-TKN-CREATEOPTIONS     token-type ZSPI-TYP-UINT.
   ZGRD-TKN-DEBUGOPTIONS      token-type ZSPI-TYP-UINT.
   ZGRD-TKN-PFS-SIZE          token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the 16-bit error code returned in the *error* parameter of PROCESS_CREATE_. This error code indicates the outcome of the process-creation attempt.

ZSPI-TKN-PROC-ERR is the procedure code. Its value is ZGRD-VAL-PROCESS-CREATE (22).

Conditional Tokens

ZGRD-TKN-XOBJECTFILE is the file name of the program file to be run.

ZGRD-TKN-XLIBRARYFILE is the name of the user library file to be used by the process, if one was explicitly named.

ZGRD-TKN-XSWAPFILE is the data swap-file name, which is passed for informational purposes only. This swap file is not used. Processes swap to a file that is managed by the Kernel-Managed Swap Facility.

ZGRD-TKN-EXTSWAPFILE is the name of the file to be used as the swap file for the default extended data stack segment of the process. It is possible that swap space is managed by the Kernel-Managed Swap Facility (KMSF), in which case, this token is not applicable. Furthermore, this token is applicable to TNS processes only.

ZGRD-TKN-PRIORITY is the priority of the new process.

ZGRD-TKN-XCPU is the processor number for the new process.

ZGRD-TKN-PROCESSHANDLE is the process handle returned by PROCESS_CREATE_. If it is null, no process was created.

ZGRD-TKN-ERRORDETAIL is the error-detail information returned by PROCESS_CREATE_ for some classes of errors.

ZGRD-TKN-NAMEOPTION is the specified process name option.

ZGRD-TKN-XPROCESSNAME is the new process name.

ZGRD-TKN-PROCESSDESCR is the process descriptor returned by PROCESS_CREATE_. If its length is 0, no process was created.

ZGRD-TKN-XHOMETERM is the name of the home terminal for the new process.

ZGRD-TKN-JOBID is the job ID to be assigned to the new process.

ZGRD-TKN-XMEMORYPAGES is the size of the data stack in pages. This token is applicable to TNS processes only.

ZGRD-TKN-CREATEOPTIONS contains the specified process-creation options.

ZGRD-TKN-DEBUGOPTIONS contains the specified debugging options.

ZGRD-TKN-PFS-SIZE the size, in bytes, of the process file segment (PFS), if a nondefault value was specified.

Effect

The effect of this error depends on the PROCESS_CREATE_ error code returned.

Recovery

Follow the recovery procedure for the returned PROCESS_CREATE_ error code as described earlier in this section.

25: ZGRD-VAL-PROCESS-LAUNCH

A call to PROCESS_LAUNCH_ failed due to an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
   ZSPI-TKN-ERRORDETAIL         token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-XOBJECTFILE         token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XLIBRARYFILE        token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XSWAPFILE           token-type ZSPI-TYP-STRING.
   ZGRD-TKN-EXTSWAPFILE         token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XPROCESSNAME        token-type ZSPI-TYP-STRING.
   ZGRD-TKN-XHOMETERM           token-type ZSPI-TYP-STRING.
   ZGRD-TKN-PFS-SIZE            token-type ZSPI-TYP-INT2.
   ZGRD-TKN-MAINSTACK-MAX       token-type ZSPI-TYP-INT2.
   ZGRD-TKN-HEAP-MAX            token-type ZSPI-TYP-INT2.
   ZGRD-TKN-SPACE-GUARANTEE     token-type ZSPI-TYP-INT2.
   ZGRD-TKN-CREATE-OPT          token-type ZSPI-TYP-INT2.
   ZGRD-TKN-NAMEOPTION          token-type ZSPI-TYP-ENUM.
   ZGRD-TKN-DEBUGOPTIONS        token-type ZSPI-TYP-UINT.
   ZGRD-TKN-PRIORITY            token-type ZSPI-TYP-UINT.
   ZGRD-TKN-XCPU                token-type ZSPI-TYP-UINT.
   ZGRD-TKN-XMEMORYPAGES        token-type ZSPI-TYP-INT.
   ZGRD-TKN-JOBID               token-type ZSPI-TYP-INT.
   ZGRD-TKN-PROCESSHANDLE       token-type ZSPI-TYP-PHANDLE.
   ZGRD-TKN-PROCESSDESCR        token-type ZSPI-TYP-STRING.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the 16-bit error code returned in the *error* parameter of PROCESS_LAUNCH_. This error code indicates the outcome of the process-creation attempt.

ZSPI-TKN-PROC-ERR is the procedure code. Its value is ZGRD-VAL-PROCESS-LAUNCH (25).

ZSPI-TKN-ERRORDETAIL is the error-detail information returned by PROCESS_LAUNCH_.

Conditional Tokens

ZGRD-TKN-XOBJECTFILE is the file name of the program file to be run.

ZGRD-TKN-XLIBRARYFILE is the name of the user library file to be used by the process, if one was explicitly named.

ZGRD-TKN-XSWAPFILE is the data swap-file name, which is passed for informational purposes only. This swap file is not used. Processes swap to a file that is managed by the Kernel-Managed Swap Facility.

ZGRD-TKN-EXTSWAPFILE is the name of the file to be used as the swap file for the default extended data stack segment of the process. It is possible that swap space is managed by the Kernel-Managed Swap Facility (KMSF), in which case, this token is not applicable. Furthermore, this token is applicable to TNS processes only.

ZGRD-TKN-XPROCESSNAME is the new process name.

ZGRD-TKN-XHOMETERM is the name of the home terminal for the new process.

ZGRD-TKN-PFS-SIZE is the size, in bytes, of the process file segment (PFS), if a nondefault value was specified.

ZGRD-TKN-MAINSTACK-MAX is the maximum size, in bytes, of the process main stack, if a nondefault value was specified.

ZGRD-TKN-HEAP-MAX is the maximum size, in bytes, of the process heap, if a nondefault value was specified. This token is applicable only to TNS/R native processes.

ZGRD-TKN-SPACE-GUARANTEE is the maximum size, in bytes, of the amount of space to be reserved with the Kernel-Managed Swap Facility (KMSF), if a nondefault value was specified.

ZGRD-TKN-CREATE-OPT contains the specified process creation options.

ZGRD-TKN-NAMEOPTION is the specified process name option.

ZGRD-TKN-DEBUGOPTIONS contains the specified debugging options.

ZGRD-TKN-PRIORITY is the initial execution priority under which the new process is to run, if a nondefault value was specified.

ZGRD-TKN-XCPU is the processor number for the new process, if a nondefault value was specified.

ZGRD-TKN-XMEMORYPAGES is the minimum number of memory pages to be allocated to the new process for user data, if a nondefault value was specified. This token is applicable only to TNS processes.

ZGRD-TKN-JOBID is the job ID to be assigned to the new process.

ZGRD-TKN-PROCESSHANDLE is the process handle associated with the new process that was returned by PROCESS_LAUNCH_. If it is null, no process was created.

ZGRD-TKN-PROCESSDESCR is the process descriptor of the new process that was returned by PROCESS_LAUNCH_. If its length is 0, no process was created.

Effect

The effect of this error depends on the PROCESS_LAUNCH_ error code returned.

Recovery

Follow the recovery procedure for the returned PROCESS_LAUNCH_ error code as described earlier in this section.

7 PROCESS_GETINFOLIST_ Errors

The PROCESS_GETINFOLIST_ procedure is used to obtain detailed information about a particular process or about processes within a processor that meet a specified list of search criteria. The process of interest can be specified either by process handle or by node name, processor number, and PIN.

The PROCESS_GETINFO_ procedure should be used when the caller requires only selected information about a particular process. PROCESS_GETINFO_ returns many of the same errors as PROCESS_GETINFOLIST_.

When PROCESS_GETINFOLIST_ returns information about one or more processes, the *error-detail* parameter contains the number of attributes returned.

Error Codes

This subsection lists each PROCESS_GETINFOLIST_ procedure error code and provides a description of each code.

```
0      NO ERROR
```

Cause. The information returned is for the processes specified; the *error-detail* parameter contains the number of processes for which information has been returned (which can be more than one process if in search mode).

Effect. None.

Recovery. None required.

```
1      A FILE SYSTEM ERROR OCCURRED
```

Cause. A file-system error occurred; the *error-detail* parameter contains the file-system error number. File-system error 563 (buffer too small) is returned if the *ret-values-list* buffer is too small to contain all of the requested information.

Effect. The procedure sets the error code and returns the error in the *error-detail* parameter.

Recovery. Refer to Section 2, File-System Errors, for corrective action.

```
2      PARAMETER ERROR
```

Cause. The PROCESS_GETINFOLIST_ call contained an illegal combination of options. The `error-detail` parameter contains the ordinal number of the first (leftmost) parameter encountered whose option is in error.

Note that parameters are counted in TAL fashion, so *nodename:length* is lumped as number 3 and *processhandle* is number 4. One reason this error can occur with detail 6 is if an attribute in *ret-value-list* specifies auxiliary data, which do not fit within the *ret-value-cnt*.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the call to PROCESS_GETINFOLIST_ so that the options are correct and in the proper order.

```
3      BOUNDS ERROR
```

Cause. There was a bounds violation on a reference parameter.

Effect. The procedure sets the error code; the `error-detail` parameter contains the number of the parameter in error.

Recovery. Pass a correct reference address to PROCESS_GETINFOLIST_.

```
4      SPECIFIED PROCESS DOES NOT EXIST
```

Cause. The specified process does not exist or does not meet the search criteria.

Effect. The information returned is for higher-numbered processes; the `error-detail` parameter contains the number of processes for which information has been returned (which can be more than one process if in search mode).

Recovery. Refer to Section 2, File-System Errors, for corrective action.

```
5      UNABLE TO COMMUNICATE WITH PROCESSOR
```

Cause. The process could not communicate with the system monitor process, possibly because the processor where the program was to be run did not exist or was inoperable. The `error-detail` parameter contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns.

Recovery. Select another processor, then try again.

```
6      UNABLE TO COMMUNICATE WITH NODE
```

Cause. The process could not communicate with the node named, possibly because the node did not exist or was inoperable. The *`error-detail`* parameter contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns.

Recovery. Select another node, then try again.

```
7      NO MORE MATCHES
```

Cause. No more matches exist.

Effect. No more matches will be found; the *`error-detail`* parameter contains the number of processes for which information has been returned (can be 0).

Recovery. None.

```
9      INVALID SEARCH CODE
```

Cause. The search attribute code was not valid; the *`error-detail`* parameter contains the invalid code.

Effect. The search is not done.

Recovery. Correct the search attribute code, then try again.

```
10      INVALID SEARCH VALUE
```

Cause. The search value was not valid; the *`error-detail`* parameter contains the associated attribute code.

Effect. The search is not done.

Recovery. Correct the attribute value, then try again.

```
11      INVALID RETURN ATTRIBUTE CODE
```

Cause. The return attribute code was not valid; the *`error-detail`* parameter contains the invalid code. This error can occur if the attribute index is unrecognized, or if a non-zero auxiliary data size is specified for an attribute that does not require auxiliary data.

Effect. The return is not done.

Recovery. Correct the return attribute code, then try again.

```
12      INVALID SEARCH OPTION
```

Cause. The search option was not valid.

Effect. The search is not done.

Recovery. Correct the search option, then try again.

```
13    - unused -
```

Cause. This number is not in current use.

Effect. N/A

Recovery. N/A

`14` Invalid auxiliary data size specification in an attribute code.

Cause. `The auxiliary data size specified was invalid.` *`error-detail`* contains the attribute code.

Effect. The procedure sets the error code and puts the attribute code in *error-detail.*

Recovery. Use the correct attribute code. The simplest way to do this is to use the attribute identifier as defined in ZSYSC or ZSYSTAL. An attribute code contains two fields: the low-order 12 bits hold the attribute index. The high-order 4 bits hold the number of 16-bit words of auxiliary data that follow the attribute code in the *ret-array-list* array. For example, attribute code ZSYS_VAL_PINF_LOADFILE_INFO is 121+4<<12, or 16505; the length value is 4 (this attribute requires eight bytes of auxiliary data).

`15` An iterative attribute was not the last attribute in *`ret-attr-list`*.

Cause. The iterative attribute specified in *error-detail* was not the last attribute in *`ret-attr-list`*.

Effect. The procedure sets the error code and puts the attribute code in *error-detail.*

Recovery. Place the iterative attribute (and its auxiliary data, if any) last in the *ret-attr-list* array.

`16` Attribute not permitted in a search request.

Cause. The attribute specified in *`error-detail`* is not permitted in a search request.

Effect. The procedure sets the error code and puts the attribute code in *error-detail.*

Recovery. Do not specify this attribute code in *et-attr-list* when the *srch-option* is 1 or 2.

`17` Attribute restricted to privileged callers.

Cause. *`error-detail`* contains the attribute code.

Effect. The procedure sets the error code and puts the attribute code in *error-detail.*

Recovery. Avoid using this attribute, or use it in a call from a function running in priviliged mode.

8 PROCESS_GETPAIRINFO_ Errors

The PROCESS_GETPAIRINFO_ procedure retrieves basic information about a process pair. The caller can specify a process handle or process descriptor, or can search through the named processes in a given system.

The PROCESS_GETINFO_ procedure should be used when the caller requires only selected information about a particular process.

Error Codes

This subsection lists each PROCESS_GETPAIRINFO_ procedure error code and provides a description of each code.

```
0      NO ERROR
```

Cause. The information returned is for the processes specified.

Effect. The information is returned about the specified process pair.

Recovery. None required.

```
2      PARAMETER ERROR
```

Cause. The PROCESS_GETPAIRINFO_ call contained an illegal combination of options.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the call to PROCESS_GETPAIRINFO_ so that the options are correct and in the proper order.

```
3      BOUNDS ERROR
```

Cause. There was a bounds violation on a reference parameter.

Effect. The procedure sets the error code.

Recovery. Pass a correct reference address to PROCESS_GETPAIRINFO_.

```
4     SINGLE NAMED PROCESS
```

Cause. A process pair was requested but only a single named process was found.

Effect. Information is returned for the single named process, which can be the caller.

Recovery. None.

```
5     CALLER'S PAIR - CALLER IS CURRENT PRIMARY
```

Cause. The information returned is for the caller's pair, and the caller is the current primary.

Effect. Information is successfully returned.

Recovery. None required.

```
6     CALLER'S PAIR - CALLER IS CURRENT BACKUP
```

Cause. The information returned is for the caller's pair, and the caller is the current backup.

Effect. Information is successfully returned.

Recovery. None required.

```
7     PROCESS IS UNNAMED
```

Cause. The process is not a named process.

Effect. No information is returned because the process is unnamed. The process can be the caller.

Recovery. No recovery action is required. Call PROCESS_GETINFO_ or PROCESS_GETINFOLIST_ to obtain the process descriptor.

```
8     SEARCH IS COMPLETE
```

Cause. The search has finished.

Effect. No information is returned. The search finished and no more information was found. Information is returned only if search by index was specified.

Recovery. None required.

```
9     PROCESS DOES NOT EXIST
```

Cause. The specified process does not exist, or the name refers to a device or volume instead of a process.

Effect. No information is returned.

Recovery. None required.

```
10     UNABLE TO COMMUNICATE WITH NODE
```

Cause. The process could not communicate with the node named, possibly because the node did not exist or was inoperable.

Effect. The procedure sets the error code and returns.

Recovery. Select another node, then try again.

```
11     TARGET IS A PROCESS CONTROLLING A DEVICE OR VOLUME,
       BUT BIT 15 WAS NOT SET
```

Cause. The target process was an IOP and bit 15 of the *options* parameter was not set.

Effect. No information is returned.

Recovery. Change the application to accept information about IOPs by changing the *options* parameter, or skip to the next processing step if the you do not want the application to process information about IOPs.

```
13     LIMITED INFORMATION IS RETURNED FOR A PROCESS THAT IS
       NOT STARTED
```

Cause. The target named process has not been started.

Effect. Limited information is returned. Returned process handles are null (-1 in each word).

Recovery. None required.

9 PROCESS_SPAWN_ Open System Services (OSS) Errors

The following error codes and error lists are produced by PROCESS_SPAWN_, the process-control procedure that creates an Open System Services (OSS) process.

The *process-results* parameter of PROCESS_SPAWN_ returns the ZSYS-DDL-PROCESSRESULTS structure, which contains errors detected by the OSS environment and errors detected by the Guardian environment. All error fields should be examined to determine the source of the error. Table 9-1 lists the error fields in the structure.

Table 9-1. PROCESS_SPAWN_ Error Fields

Field	Description
Z-ERRNO	is an OSS `errno` value that is returned by the OSS environment.
Z-TPCERROR	is a process creation error that is returned by the Guardian environment.
Z-TPCDETAIL	is a value that provides further detail on some process creation errors reported in Z-TPCERROR.

If PROCESS_SPAWN_ is called in a nowait manner, error information is returned in system message -141 (nowait PROCESS_SPAWN_ completion). See Section 20, System Messages, for details.

For further information about starting processes and about the PROCESS_SPAWN_ procedure, refer to the *Guardian Procedure Calls Reference Manual*.

Guardian Error Codes

The PROCESS_SPAWN_ Guardian error codes are the values returned to the calling process in the ZSYS-DDL-PROCESSRESULTS.Z-TPCERROR field of the *process-results* parameter.

For many classes of Guardian errors, additional information is returned in the ZSYS-DDL-PROCESSRESULTS.Z-TPCDETAIL field of the *process-results* parameter.

Guardian process creation error codes returned by PROCESS_SPAWN_ are also returned by PROCESS_CREATE_. These error codes are described in Section 6, Process Creation Errors.

OSS Error Codes

The PROCESS_SPAWN_ OSS error codes are the values returned to the calling process in the ZSYS-DDL-PROCESSRESULTS.Z-ERRNO field of the *`process-results`* parameter. Commonly returned OSS error codes are described in this section. For more information on OSS error codes, see Section 22, OSS Error Information.

```
0      NO ERROR
```

Cause. The PROCESS_SPAWN_ call was completed successfully. The corresponding OSS `errno` value is `ENOERR`.

Effect. The process was created, or creation was initiated if PROCESS_SPAWN_ was called in a nowait manner. In the latter case, creation results are returned in a user-level system message.

Recovery. Informative message only; no corrective action is needed.

```
4002     NO SUCH FILE OR DIRECTORY
```

Cause. Either the pathname is empty or a component of a specified pathname does not exist. The corresponding OSS `errno` value is `ENOENT`.

Effect. No process is created.

Recovery. Create a file with the specified pathname or specify a pathname that exists.

```
4005     I/O ERROR
```

Cause. A physical input or output error occurred during an attempt to create or access a file. The corresponding OSS `errno` value is `EIO`.

Effect. No process is created.

Recovery. Corrective action is application dependent.

```
4007     ARGUMENT LIST TOO LONG
```

Cause. The number of bytes available for the new process's combined argument (*`argv`*) and environment (*`envp`*) lists has exceeded the system-imposed limit. This limit, which includes the pointers and the null terminators on the strings, is available by calling the `sysconf(_SC_ARG_MAX)` OSS function. The corresponding OSS `errno` value is `E2BIG`.

Effect. No process is created.

Recovery. Specify a shorter argument list in the *`argv`* parameter and a shorter environment list in the *`envp`* parameter.

```
4008     EXEC FORMAT ERROR
```

Cause. The OSS program file specified by the *oss-program-file* parameter has the appropriate permissions, but it is not in the format for executable files. The corresponding OSS `errno` value is `ENOEXEC`.

Effect. No process is created.

Recovery. Either correct the *oss-program-file* parameter to refer to an executable program file or alter the program file such that it can be executed. The program file must be either a valid object file with a SYSTYPE of OSS or a valid script file.

```
4009     BAD FILE DESCRIPTOR
```

Cause. A file descriptor specified in the *fdinfo* parameter is invalid. The corresponding OSS `errno` value is `EBADF`.

Effect. No process is created.

Recovery. Correct the *fdinfo* parameter.

```
4011     NO MORE PROCESSES
```

Cause. System resources such as the process control block (PCB) space, MAPPOOL space, stack space, or process file segment (PFS) space are inadequate. It is also possible that the process name selected is already in use. The corresponding OSS `errno` value is `EAGAIN`.

Effect. No process is created.

Recovery. Check the system for processes that are using too much memory, terminate processes that are no longer needed, and call PROCESS_SPAWN_ again. Check the program to see whether it uses too much buffer space, opens too many files, or uses too many DEFINEs. If the problem is with the PFS, try running the process with a larger PFS either by specifying a larger PFS size in the ZSYS^DDL^PROCESSEXTENSION.Z^PFSSIZE field of the process-extension parameter or by setting the size with the Binder. If the process name selected is already in use, then choose another process name and call PROCESS_SPAWN_ again.

```
4012     INSUFFICIENT USER MEMORY
```

Cause. There is insufficient user memory to create the OSS process. The new process image requires more memory than is allowed by the hardware or system-imposed memory management constraints. The corresponding OSS `errno` value is `ENOMEM`.

Effect. No process is created.

Recovery. Terminate OSS processes that are no longer needed and call PROCESS_SPAWN_ again.

```
4013     PERMISSION DENIED
```

Cause. Search permission is denied on the pathname or a component of a pathname prefix. The corresponding OSS `errno` value is `EACCES`.

Effect. No process is created.

Recovery. Correct the parameter containing the erroneous pathname.

```
4014     BAD ADDRESS
```

Cause. A specified parameter has an invalid address. The corresponding OSS `errno` value is `EFAULT`.

Effect. No process is created.

Recovery. Correct the erroneous parameter.

```
4020     NOT A DIRECTORY
```

Cause. A prefix within a specified pathname refers to a file that is not a directory. The corresponding OSS `errno` value is `ENOTDIR`.

Effect. No process is created.

Recovery. Correct the parameter containing the erroneous pathname.

```
4022     INVALID FUNCTION ARGUMENT
```

Cause. One of the specified parameters is not valid or a required parameter is not specified. The corresponding OSS `errno` value is `EINVAL`.

Effect. No process is created.

Recovery. Correct the erroneous parameter.

```
4126     CONNECTION TIMED OUT
```

Cause. The operation timed out. The timeout value specified in the *`fdinfo`* parameter was reached before the file descriptors could be opened. The corresponding OSS `errno` value is `ETIMEDOUT`.

Effect. No process is created.

Recovery. Call the PROCESS_SPAWN_ procedure again or increase the timeout value.

```
4131     FILE NAME TOO LONG
```

Cause. The specified pathname or a component of the pathname is longer than `PATH_MAX` characters. (`PATH_MAX` is a symbolic constant that is defined in the OSS `limits.h` header file.) The corresponding OSS `errno` value is `ENAMETOOLONG`.

Effect. No process is created.

Recovery. Correct the parameter containing the erroneous pathname.

```
4203     OSS NOT RUNNING OR NOT INSTALLED
```

Cause. Open System Services is not running. To create an OSS process, Open System Services must be running. The corresponding OSS `errno` value is `EOSSNOTRUNNING`.

Effect. No process is created.

Recovery. Run Open System Services.

```
4212     AN ERROR OCCURRED DURING INVOCATION OF A DEFINE
         PROCEDURE
```

Cause. An error exists in a Guardian DEFINE. The corresponding OSS `errno` value is `EDEFINEERR`.

Effect. No process is created.

Recovery. Recovery is application dependent. See the *Guardian Programmer’s Guide* for an explanation of how to use DEFINEs in procedure calls.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a PROCESS_SPAWN_ error list in a response. HP subsystems return such an error list when, in performing your request, they call the PROCESS_SPAWN_ procedure directly or indirectly and an error occurs on the call.

The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every PROCESS_SPAWN_ error list, identifies the procedure. Its value is ZGRD-VAL-PROCESS-SPAWN (24).

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a PROCESS_SPAWN_ error list:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in the error-list description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about tokens and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

24: ZGRD-VAL-PROCESS-SPAWN

There was an error during a call to PROCESS_SPAWN_.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST               token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
   ZSPI-TKN-PROCESS-RESULTS    token-type ZSPI-TYP-BYTESTRING.
ZSPI-TKN-ENDLIST               token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-PROGRAM-PATH       token-type ZSPI-TYP-STRING.
   ZGRD-TKN-LIBRARY-PATH       token-type ZSPI-TYP-STRING.
   ZGRD-TKN-SWAP-PATH          token-type ZSPI-TYP-STRING.
   ZGRD-TKN-EXTSWAP-PATH       token-type ZSPI-TYP-STRING.
   ZGRD-TKN-TERM-PATH          token-type ZSPI-TYP-STRING.
   ZGRD-TKN-CWD-PATH           token-type ZSPI-TYP-STRING.
   ZGRD-TKN-FD-PATH            token-type ZSPI-TYP-STRING.
   ZGRD-TKN-FDINFO             token-type ZSPI-TYP-BYTESTRING.
   ZGRD-TKN-ARG                token-type ZSPI-TYP-STRING.
   ZGRD-TKN-ENV                token-type ZSPI-TYP-STRING.
   ZGRD-TKN-PROCESS-EXTENSION token-type ZSPI-TYP-BYTESTRING.
   ZGRD-TKN-PATH               token-type ZSPI-TYP-STRING.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the OSS `errno` returned in the ZSYS-DDL-PROCESSRESULTS.Z-ERRNO field of the *process-results* parameter of PROCESS_SPAWN_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZGRD-VAL-PROCESS-SPAWN (24).

ZSPI-TKN-PROCESS-RESULTS is the value of the *process-results* parameter (containing the structure ZSYS-DDL-PROCESSRESULTS) of PROCESS_SPAWN_.

Conditional Tokens

ZGRD-TKN-PROGRAM-PATH is the OSS pathname of the program file.

ZGRD-TKN-LIBRARY-PATH is the OSS pathname of the library file.

ZGRD-TKN-SWAP-PATH is the OSS pathname of the swap file.

ZGRD-TKN-EXTSWAP-PATH is the OSS pathname of the extended swap file.

ZGRD-TKN-TERM-PATH is the OSS pathname of the home terminal for the new process.

ZGRD-TKN-CWD-PATH is the OSS pathname of the current working directory for the new process.

ZGRD-TKN-FD-PATH is the OSS pathname of a file descriptor to be opened by the new process. The corresponding pointer to this OSS pathname is in the ZSYS-DDL-FDINFO.Z-FDENTRY.Z-NAME field of the *fdinfo* parameter. The *fdinfo* parameter is in the ZGRD-TKN-FDINFO token. If the pointer is null (0D), then a zero-length OSS pathname must be specified. Because there can be multiple file descriptors to be opened, there can be multiple occurrences of the Z-FDENTRY substructure and of the corresponding ZGRD-TKN-FD-PATH token. Multiple ZGRD-TKN-FD-PATH tokens must be provided in the same order as the pointers to the OSS pathnames in the *fdinfo* parameter.

ZGRD-TKN-FDINFO is the value of the *fdinfo* parameter.

ZGRD-TKN-ARG is an argument string pointed to by the *argv* parameter. The *argv* parameter is an array of pointers to strings. Because there can be multiple strings, there can be multiple occurrences of the ZGRD-TKN-ARG token. Multiple ZGRD-TKN-ARG tokens must be provided in the same order as elements of the *argv* array.

ZGRD-TKN-ENV is the process environment string pointed to by the *envp* parameter. The *envp* parameter is an array of pointers to strings. Because there can be multiple strings, there can be multiple occurrences of the ZGRD-TKN-ENV token. The ZGRD-TKN-ENV tokens do not need to be provided in the same order as the elements of the *envp* array.

ZGRD-TKN-PROCESS-EXTENSION is the value of the *process-extension* parameter.

ZGRD-TKN-PATH is the value of the *path* parameter.

Effect

The effect of this error depends on the values returned in the ZSPI-TKN-PROCESS-RESULTS token.

Recovery

Follow the recovery procedure for the returned PROCESS_SPAWN_ errors as described earlier in this section.

10 ALLOCATESEGMENT Errors

This section contains the error codes returned by the ALLOCATESEGMENT procedure and the error lists associated with the ALLOCATESEGMENT procedure. Error codes and error lists associated with the SEGMENT_ALLOCATE_ procedure are described in Section 11, SEGMENT_ALLOCATE_ Errors. The error list for USESEGMENT is described in Section 12, USESEGMENT Errors. The error list for SEGMENT_USE_ is described in Section 13, SEGMENT_USE_ Errors.

Error Codes

The ALLOCATESEGMENT error codes are the values returned in the *status* parameter to ALLOCATESEGMENT. ALLOCATESEGMENT error codes in the range 1 through 999 indicate a file-system error related to the creation or opening of a swap file. Refer to Section 2, File-System Errors, for information about these codes.

```
0        NO ERROR
```

Cause. The operation was completed successfully.

Effect. None.

Recovery. Informative message only; no corrective action is needed.

```
-1       ILLEGAL SEGMENT ID
```

Cause. The ALLOCATESEGMENT call either specified an invalid extended data segment ID (not in the range 0 through 1023) or used the extended data segment ID of a currently allocated extended data segment.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid extended data segment ID, then retry the operation. User processes can specify an extended data segment ID equal to 1023 or less; only processes supplied by HP can specify an extended data segment ID greater than 1023.

```
-2       ILLEGAL SEGMENT SIZE
```

Cause. The ALLOCATESEGMENT call specified an invalid extended data segment size.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid extended data segment size, then retry the operation. The range of valid values for a selectable segment is between 1 byte and 127.5

megabytes. (To allocate a shared flat segment, you must use the SEGMENT_ALLOCATE_ procedure.)

```
-3       BOUNDS VIOLATION ON SWAP FILE PARAMETER
```

Cause. A bounds violation occurred on the ALLOCATESEGMENT swap-file name.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Pass a correct reference address or the swap-file name to ALLOCATESEGMENT.

```
-4       ILLEGAL COMBINATION OF OPTIONS
```

Cause. The ALLOCATESEGMENT call contained an illegal combination of options.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Correct the call to ALLOCATESEGMENT so that the options are correct and in the proper order.

```
-5       UNABLE TO ALLOCATE SEGMENT SPACE
```

Cause. ALLOCATESEGMENT could not allocate extended data segment space.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Retry the operation, or run the application on another processor.

```
-6       UNABLE TO ALLOCATE SEGMENT PAGE TABLE SPACE
```

Cause. ALLOCATESEGMENT could not allocate any segment page table space. The segment page table stores one entry for each page allocated in the extended data segment that it is related to.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Retry the operation, or run the application on another processor.

```
-7       SECURITY VIOLATION
```

Cause. A security violation occurred when the process tried to share extended data segment space using ALLOCATESEGMENT.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Make sure the calling process access ID is one of the following:

- The same as that of the process whose PIN is specified in the ALLOCATESEGMENT call
- The group manager for the access ID of the other process
- The super ID (255,255)

```
-8       PIN DOES NOT EXIST
```

Cause. The ALLOCATESEGMENT call specified an invalid process identification number (PIN).

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid PIN, then retry the operation.

```
-9       NO SEGMENT ALLOCATED TO PIN
```

Cause. The process specified for segment sharing has not allocated an extended data segment or the ID does not match the requestor.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid PIN, then retry the operation.

```
-10      TRYING TO SHARE SEGMENT WITH SELF
```

Cause. The process calling ALLOCATESEGMENT is trying to share an extended data segment with itself.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Make sure the PIN specified in *pin-and-flags* is not the PIN of the calling process.

```
-11       REQUESTED SEGMENT IS CURRENTLY BEING RESIZED
```

Cause. Either the requested segment cannot be accessed by SEGMENT_ALLOCATE_ until the current resizing operation is complete or the requested segment is a shared selectable segment but the allocated segment is a flat segment.

Effect. The procedure sets the error code and returns without accessing the extended data segment.

Recovery. Wait until the current resizing operation is complete, then retry the operation. Make sure that the requested segment and the allocated segment are either both flat segments or both selectable segments when sharing.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive an ALLOCATESEGMENT error list in a response. Subsystems return such an error list when, in performing your request, they call the ALLOCATESEGMENT procedure directly or indirectly, and an error occurs on the call.

The contents of the error list depend on which procedure was called. The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every ALLOCATESEGMENT error list, identifies the procedure. Its value is ZGRD-VAL-ALLOCATESEGMENT (1).

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing an ALLOCATESEGMENT error list:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in each error-list description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about tokens and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

1: ZGRD-VAL-ALLOCATESEGMENT

A call to ALLOCATESEGMENT returned a nonzero status as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-OBJECTFILE     token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-SEGMENTID      token-type ZSPI-TYP-INT.
   ZGRD-TKN-SEGMENTSIZE    token-type ZSPI-TYP-INT2.
   ZGRD-TKN-FILENAME       token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-PINANDFLAGS    token-type ZSPI-TYP-BYTE-PAIR.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the error code returned in the *status* parameter of ALLOCATESEGMENT.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZGRD-VAL-ALLOCATESEGMENT (1).

Conditional Tokens

ZGRD-TKN-OBJECTFILE is the name of the object file containing the ALLOCATESEGMENT procedure call in internal format.

ZGRD-TKN-SEGMENTID is the extended data segment ID requested.

ZGRD-TKN-SEGMENTSIZE is the extended data segment size in bytes.

ZGRD-TKN-FILENAME is the swap file name.

ZGRD-TKN-PINANDFLAGS contains the value of the *pin-and-flags* parameter:

Value	**Meaning**
PINANDFLAGS.<0:7>	Flag options
PINANDFLAGS.<8:15>	PIN of process sharing the extended data segment

Effect

The extended data segment is not allocated.

Recovery

Follow the recovery procedure for the returned ALLOCATESEGMENT error status as described earlier in this section.

11 SEGMENT_ALLOCATE_ Errors

This section lists and describes the error codes and error lists associated with the SEGMENT_ALLOCATE_ procedure. The SEGMENT_ALLOCATE_ error codes are the values returned in the *error* parameter to SEGMENT_ALLOCATE_. The same error codes are returned by SEGMENT_ALLOCATE_CHKPT_.

Error codes and error lists associated with the ALLOCATESEGMENT procedure are described in Section 10, ALLOCATESEGMENT Errors. The error list for USESEGMENT is described in Section 12, USESEGMENT Errors. The error list for SEGMENT_USE_ is described in Section 13, SEGMENT_USE_ Errors.

Error Codes

This subsection lists each SEGMENT_ALLOCATE_ procedure error code and provides a description of each code.

```
0        NO ERROR
```

Cause. The operation was completed successfully.

Effect. None.

Recovery. Informative message only; no corrective action is needed.

```
1        ERROR CREATING OR OPENING SEGMENT SWAP FILE
```

Cause. The system encountered an error while creating or opening the swap file. The *error-detail* parameter contains the error number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Correct the call to SEGMENT_ALLOCATE_ to make sure the correct swap file is specified. If there is no disk space available to create the swap file, either specify a different volume or create space on the desired volume.

```
2       PARAMETER ERROR
```

Cause. The SEGMENT_ALLOCATE_ call contained an illegal combination of options. Possible causes include:

- Omitting the required *segment-id* parameter
- Supplying a *max-size* value that is inconsistent with *segment-size*
- Supplying an invalid *segment-type*
- Setting one or more “reserved, must be 0” bits to a nonzero value

The *error-detail* parameter contains the ordinal number of the first (leftmost) parameter encountered whose option is in error.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Correct the call to SEGMENT_ALLOCATE_ so that the options are correct and in the proper order.

```
3       BOUNDS VIOLATION
```

Cause. A bounds violation occurred on a reference parameter. The reference parameter address is checked for length and location. The *error-detail* parameter contains the ordinal number of the first (leftmost) parameter encountered whose option is in error.

Effect. The procedure sets the error code and returns without any further action.

Recovery. Pass a correct reference address to SEGMENT_ALLOCATE_.

```
4       ILLEGAL SEGMENT ID
```

Cause. The SEGMENT_ALLOCATE_ call either specified an invalid segment ID (not in the range 0 through 1023) or used the segment ID of a currently allocated segment.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid segment ID, then retry the operation. User processes can specify a segment ID equal to 1023 or less; only processes supplied by HP can specify a segment ID greater than 1023.

```
5       ILLEGAL SEGMENT SIZE
```

Cause. The SEGMENT_ALLOCATE_ call specified an invalid segment size.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid segment size, then retry the operation. Valid values for *segment-size* depend on the type of segment to be allocated:

- For a flat segment, *segment-size* must be in the range of 1 byte to 128 megabytes.
- For a selectable segment, *segment-size* must be in the range of 1 byte to 127.5 megabytes.

```
6       UNABLE TO ALLOCATE SEGMENT SPACE
```

Cause. SEGMENT_ALLOCATE_ could not allocate segment space because not enough contiguous memory was available.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Retry the operation, or run the application on another processor.

```
7       UNABLE TO ALLOCATE SEGMENT PAGE TABLE SPACE
```

Cause. SEGMENT_ALLOCATE_ could not allocate any segment page table space. The segment page table stores one entry for each page allocated in the segment that it is related to.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Retry the operation, or run the application on another processor.

```
8       SECURITY VIOLATION
```

Cause. A security violation occurred when the process tried to share segment space using SEGMENT_ALLOCATE_. This error indicates that the segment space could not be shared.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Make sure the calling process access ID is one of the following:

- The same as the process whose process identification number (PIN) is specified in the SEGMENT_ALLOCATE_ call.
- The group manager for the access ID of the other process
- The super ID (255,255)

```
9       PIN DOES NOT EXIST
```

Cause. The SEGMENT_ALLOCATE_ call specified an invalid PIN.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid PIN, then retry the operation.

```
10      NO SEGMENT ALLOCATED TO PIN
```

Cause. The process specified for extended data segment sharing has not allocated a segment or IDs do not match.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Specify a valid PIN, then retry the operation.

```
11      TRYING TO SHARE SEGMENT WITH SELF
```

Cause. The process calling SEGMENT_ALLOCATE_ is trying to share an extended data segment with itself.

Effect. The procedure sets the error code and returns without allocating the extended data segment.

Recovery. Make sure the PIN specified in *pin* is not the calling process's PIN.

```
12      REQUESTED SHARED SEGMENT IS INCOMPATIBLE
```

Cause. The requested segment is a shared selectable segment but the allocated segment is a flat segment; or the requested segment is a shared flat segment but the allocated segment is a selectable segment.

Effect. The procedure sets the error code and returns without accessing the extended data segment.

Recovery. Make sure that the requested segment and the allocated segment are either both flat segments or both selectable segments when sharing.

```
13      REQUESTED SEGMENT ALREADY ALLOCATED
```

Cause. The requested segment is already allocated for this process.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Specify an unused segment ID, then retry the operation.

```
14     UNABLE TO ALLOCATE THE PST
```

Cause. The requested process segment table (PST) cannot be allocated. The *error-detail* parameter contains the number of the file-system error that occurred.

Effect. The procedure sets the error code and returns without accessing the extended data segment.

Recovery. Wait until more process file segment (PFS) space is available, or run the application on another processor. Refer to Section 2, File-System Errors, for information about the file-system error returned in *error-detail*.

```
15      REQUESTED ADDRESS RANGE ALREADY ALLOCATED
```

Cause. Part or all of the specified address range has already been allocated. This error is returned if bit 15 of the alloc-options parameter is set to 1 and a flat segment cannot be allocated. This error can also occur when bit 15 is not set, but either a flat segment cannot be shared due to address-range overlap with another segment or a flat segment cannot be allocated as there is no unallocated address range large enough to hold the requested size. This error is returned only on TNS/R processors.

Effect. The extended data segment cannot be allocated.

Recovery. Correct the specification and try again.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a SEGMENT_ALLOCATE_ error list in a response. HP subsystems return such an error list when, in performing your request, they call the SEGMENT_ALLOCATE_ procedure directly or indirectly and an error occurs on the call.

The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every SEGMENT_ALLOCATE_ error list, identifies the procedure. Its value is ZGRD-VAL-SEGMENT-ALLOCATE (21).

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a SEGMENT_ALLOCATE_ error list:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in the error-list description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about token and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

21: ZGRD-VAL-SEGMENT-ALLOCATE

A call to SEGMENT_ALLOCATE_ returned a nonzero error as the function value.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR       token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-XOBJECTFILE    token-type ZSPI-TYP-STRING.
   ZGRD-TKN-SEGMENTID      token-type ZSPI-TYP-INT.
   ZGRD-TKN-SEGMENTSIZE    token-type ZSPI-TYP-INT2.
   ZGRD-TKN-XFILENAME      token-type ZSPI-TYP-STRING.
   ZGRD-TKN-ERRORDETAIL    token-type ZSPI-TYP-INT.
   ZGRD-TKN-PIN            token-type ZSPI-TYP-UINT.
   ZGRD-TKN-SEGMENTTYPE    token-type ZSPI-TYP-ENUM.
   ZGRD-TKN-MAXSEGSIZE     token-type ZSPI-TYP-INT2.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the error code returned in the *error* parameter of SEGMENT_ALLOCATE_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZGRD-VAL-SEGMENT-ALLOCATE (21).

Conditional Tokens

ZGRD-TKN-XOBJECTFILE is the name of the object file that contains the SEGMENT_ALLOCATE_ procedure call in internal format.

ZGRD-TKN-SEGMENTID is the extended segment ID requested.

ZGRD-TKN-SEGMENTSIZE is the extended segment size in bytes.

ZGRD-TKN-XFILENAME is the swap-file name in internal format.

ZGRD-TKN-ERRORDETAIL is the detailed information associated with some classes of errors.

ZGRD-TKN-PIN is the process identification number (PIN) of the process sharing the segment.

ZGRD-TKN-SEGMENTTYPE is the type of segment requested.

ZGRD-TKN-MAXSEGSIZE is the maximum size that the segment can grow to using calls to RESIZESEGMENT.

Effect

The extended segment is not allocated.

Recovery

Follow the recovery procedure for the returned SEGMENT_ALLOCATE_ error status as described earlier in this section.

12 USESEGMENT Errors

This section contains the error list associated with the USESEGMENT procedure. There are no error codes associated with USESEGMENT; USESEGMENT does not return error codes.

Note that the USESEGMENT procedure is similar to the SEGMENT_USE_ procedure. You would typically use the SEGMENT_USE_ procedure on D-series and later releases, while you used the USESEGMENT procedure on C-series releases. See Section 13, SEGMENT_USE_ Errors, for the error list associated with the SEGMENT_USE_ procedure.

Also see Section 10, ALLOCATESEGMENT Errors, for error codes and error lists associated with the ALLOCATESEGMENT procedure.   See Section 11, SEGMENT_ALLOCATE_ Errors, for error codes and error lists associated with the SEGMENT_ALLOCATE_ procedure.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a USESEGMENT error list in a response. Subsystems return such an error list when, in performing your request, they call the USESEGMENT procedure directly or indirectly and an error occurs on the call.

The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every USESEGMENT error list, identifies the procedure. Its value is ZGRD-VAL-USESEGMENT (2).

The error list always includes the unconditional tokens listed under its description in this section. In addition, the error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a USESEGMENT error list:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in the error-list description.

This section does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about tokens and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

2: ZGRD-VAL-USESEGMENT

A condition code less (CCL) was returned by a call to USESEGMENT. Either the segment ID in ZGRD-TKN-SEGMENTID is not allocated or it cannot be used by a nonprivileged caller.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-OBJECTFILE       token-type ZSPI-TYP-FNAME.
   ZGRD-TKN-SEGMENTID        token-type ZSPI-TYP-INT.
   ZGRD-TKN-OLDSEGMENT       token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the error code returned in the *status* parameter of USESEGMENT.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZGRD-VAL-USESEGMENT (2).

Conditional Tokens

ZGRD-TKN-OBJECTFILE is the name of the object file that contains the USESEGMENT procedure call in internal format.

ZGRD-TKN-SEGMENTID is the extended data segment ID requested.

ZGRD-TKN-OLDSEGMENT is the extended data segment ID returned.

Effect

The extended data segment requested is not enabled for use by the caller.

Recovery

Make sure the specified extended data segment ID is already allocated and that the segment ID is 1023 or less.

13 SEGMENT_USE_ Errors

This section contains the error list associated with the SEGMENT_USE_ procedure. This procedure selects a particular extended data segment to be currently addressable by the calling process.

Note that the SEGMENT_USE_ procedure is similar to the USESEGMENT procedure. You would typically use the SEGMENT_USE_ procedure on D-series and later releases, while you used the USESEGMENT procedure on C-series releases. See Section 12, USESEGMENT Errors for the error list associated with the USESEGMENT procedure.

Also see Section 10, ALLOCATESEGMENT Errors, for error codes and error lists associated with the ALLOCATESEGMENT procedure.   See Section 11, SEGMENT_ALLOCATE_ Errors, for error codes and error lists associated with the SEGMENT_ALLOCATE_ procedure.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a SEGMENT_USE_ error list in a response. Subsystems return such an error list when, in performing your request, they call the SEGMENT_USE_ procedure directly or indirectly and an error occurs on the call.

The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every SEGMENT_USE_ error list, identifies the procedure. Its value is ZGRD-VAL-SEGMENT-USE (23).

The error list always includes the unconditional tokens listed under its description in this section. In addition, the error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing a SEGMENT_USE_ error list:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in the error-list description.

This section does not discuss the mechanics of error-list construction. For information about creating error lists, additional information about tokens and token types, and definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

23: ZGRD-VAL-SEGMENT-USE

A call to SEGMENT_USE_ returned a nonzero error as the function value. There are several possible causes of the error, as explained below.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST               token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR              token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR           token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST               token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-XOBJECTFILE        token-type ZSPI-TYP-STRING.
   ZGRD-TKN-SEGMENTID          token-type ZSPI-TYP-INT.
   ZGRD-TKN-OLDSEGMENT         token-type ZSPI-TYP-INT.
   ZGRD-TKN-ERRORDETAIL        token-type ZSPI-TYP-INT.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the error code returned in the *error* return value of SEGMENT_USE_.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZGRD-VAL-SEGMENT-USE (23).

Conditional Tokens

ZGRD-TKN-XOBJECTFILE is the name of the object file that contains the SEGMENT_USE_ procedure call in internal format.

ZGRD-TKN-SEGMENTID is the extended data segment ID requested.

ZGRD-TKN-OLDSEGMENT is the previous extended data segment ID in use.

ZGRD-TKN-ERRORDETAIL is the detailed information associated with some classes of errors.

Effect

The extended data segment requested is not enabled for use by the caller.

Recovery

The recovery procedure depends on the specific error code returned in the *error* return value.

- If the error code is 2 (parameter error), the SEGMENT_USE_ call contained an illegal combination of options. Specifically, the required *new-segid* parameter was omitted. Correct the SEGMENT_USE_ call by providing the *new-segid* parameter.
- If the error code is 3 (bounds violation), a bounds violation occurred on a reference parameter. To recover from this error, pass a correct reference address to SEGMENT_USE_.
- If the error code is 4 (illegal segment ID), the SEGMENT_USE_ call specified a segment ID (in the *new-segid* parameter) that is illegal for an unprivileged caller. That is, the segment ID is not in the range 0 through 2047. To recover from this error, specify a valid segment ID and then retry the operation. User processes can specify a segment ID equal to 2047 or less. Only processes supplied by HP can specify a segment ID greater than 2047.
- If the error code is 5 (segment ID not found), the SEGMENT_USE_ call specified a segment ID (in the *new-segid* parameter) that has not been allocated. To recover from this error, specify a valid, allocated segment ID and then retry the operation.

14 Subsystem Programmatic Interface (SPI) Errors

This section lists and describes the error numbers issued by the Subsystem Programmatic Interface (SPI) procedures and the error lists associated with those procedures.

Error Codes

The following pages describe the SPI error numbers. These numbers are the values returned in the *status* parameter on calls to the SPI procedures.

When any of these errors (except error 0 or error -1) occurs, the header token ZSPI-TKN-LASTERR is set to the error number.

```
0       ZSPI-ERR-OK             No error
```

Cause. The operation completed successfully.

Effect. The requested operation is complete.

Recovery. Informative message only; no corrective action is needed.

```
-1       ZSPI-ERR-INVBUF         Invalid buffer format
```

Cause. The buffer supplied in the procedure call has an invalid format, as indicated by one or more of the following:

- The first word of the buffer does not contain the correct message code (ZSPI-VAL-MSGCODE = -28).
- The buffer is in a format not recognized by the current version of SPI.
- The used length of the buffer (ZSPI-TKN-USEDLEN) is greater than the maximum length (Z-BUFLEN). Probably SSPUT was called with ZSPI-TKN-RESET-BUFFER and a *maxlen* value that is smaller than ZSPI-TKN-USEDLEN.
- The buffer contains a ZSPI-TKN-ENDLIST token but no corresponding list token.
- The position descriptor within the buffer (ZSPI-TKN-POSITION) indicates a current list that does not begin with a list token. Perhaps an incorrect position descriptor—one saved from another buffer—was restored to ZSPI-TKN-POSITION in this buffer.

Effect. The requested operation is not completed. Since the buffer format is invalid, the last error is not saved.

Recovery. Check for an incorrect *buffer* parameter or a corrupted buffer.

```
-2        ZSPI-ERR-ILLPARM        Illegal parameter value
```

Cause. A parameter supplied in the procedure call was illegal for one of the following reasons:

- SSINIT was called with an invalid header type.
- A negative *index* or *count* parameter was supplied.
- An attempt was made to use SSPUT or SSGET on a token using a *count* of zero.
- A call was made to one of the special operations that returns attributes of a token, but the operation was applied to one of the special tokens. The special tokens that return attributes are ZSPI-TKN-COUNT, ZSPI-TKN-LEN, ZSPI-TKN-OFFSET, and ZSPI-TKN-ADDR.
- The program supplied an SPI-defined token code that was invalid for this procedure call. For example, ZSPI-TKN-DELETE was specified in a call to SSGET, or ZSPI-TKN-COMMAND was specified in a call to SSPUT.
- An invalid position descriptor was supplied with ZSPI-TKN-POSITION.

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Correct the parameter in error.

```
-3       ZSPI-ERR-MISPARM        Missing parameter
```

Cause. This error number indicates that a required parameter was not supplied. Certain parameters are required only under certain circumstances:

- The *ssid* parameter is required when calling SSGET with ZSPI-TKN-NEXTCODE or ZSPI-TKN-NEXTTOKEN if the next token code in the buffer is not qualified by the default subsystem ID. Always supply a variable for *ssid* when calling SSGET with ZSPI-TKN-NEXTCODE or ZSPI-TKN-NEXTTOKEN unless you are certain that all tokens the program could encounter are qualified by the default subsystem ID.
- The *value* parameter is required when calling SSGET with certain standard token codes (such as ZSPI-TKN-LEN and ZSPI-TKN-OFFSET) or when calling SSPUT with a token code that has a value (a nonzero token length).

Effect. The header token ZSPI-TKN-LASTERR is set to this error number (unless the *buffer* parameter is missing), and the requested operation is not performed.

Recovery. Supply the missing parameter.

```
-4       ZSPI-ERR-BADADDR        Illegal parameter address
```

Cause. A reference parameter has an illegal address for one of the following reasons:

- A parameter has a starting address that is invalid or out of bounds.
- A parameter has an absolute extended address and the caller is nonprivileged.
- A parameter's starting address and length are such that the parameter overlaps the stack space that is required by the SPI procedure.
- An extended-address parameter refers to the current code space. This can occur if the parameter is a read-only array located in a user-library space.

Effect. The header token ZSPI-TKN-LASTERR is set to this error number (unless the bounds error occurs on the *buffer* parameter), and the requested operation is not performed.

If the bounds error occurs on the *buffer* parameter, SPI is unable to find the buffer, so it cannot set ZSPI-TKN-LASTERR.

Recovery. Correct the parameter declarations to allocate the required storage.

```
-5       ZSPI-ERR-NOSPACE        Buffer full
```

Cause. This error can occur for one of the following reasons:

- The buffer is full; it cannot contain any more tokens or header information.
- SSPUT was called with ZSPI-TKN-RESET-BUFFER, but *maxlen* was smaller than the used length of the buffer. In this case, some information at the end of the message was lost. Subsequent SPI calls for this buffer will return error -1 (invalid buffer format).

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. For the first cause, use a larger buffer. For the second cause, recovery is application-dependent.

```
-6       ZSPI-ERR-XSUMERR        Invalid checksum
```

Cause. The current buffer checksum does not match the checksum computed on return from the last SPI call. This error suggests that the buffer has been modified or damaged.

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Using a debugging tool such as the Inspect debugger, check for corruption of the buffer contents.

```
-7       ZSPI-ERR-INTERR         Internal error
```

Cause. This internal error should not occur unless the SPI software malfunctions. Specific causes include:

- SSGET attempted to return a token value when the program had requested a token attribute (such as length, offset, address, or count).
- SSGET attempted to return an undefined token attribute.
- On returning, SSGET or SSPUT attempted to set a used length greater than the buffer length.
- SSPUT received an error when calling SSGET to obtain the default subsystem ID (ZSPI-TKN-DEFAULT-SSID) from the SPI message header.

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Report the problem to your service provider, supplying a reproducible test case.

```
-8       ZSPI-ERR-MISTKN         Token not found
```

Cause. This error can occur for one of the following reasons:

- The token requested in a call to SSGET was not found.
- An attempt was made to put a ZSPI-TKN-ENDLIST token into the buffer, but no corresponding list token was found.

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Corrective action depends on the application. Check for program logic errors in scanning the buffer. Also check to see whether the token is positioned outside the current list or preceding the current position.

```
-9       ZSPI-ERR-ILLTKN         Illegal token code or map
```

Cause. An illegal token code or token map was supplied in the procedure call. Possible causes for this error include:

- The token data type was not recognized. For example, a program used a token data type not included in the standard SPI definitions. The only token data types permitted by SPI are those defined by SPI.
- The token length was not a multiple of the basic length associated with the token data type.
- The token map contained an invalid null-value specification.
- The sum of the lengths of the null-value specifications in the token map was not equal to the total structure length specified by the map.
- A token map was supplied as a parameter to SSGETTKN, SSPUTTKN, or SSMOVETKN.

Any of these situations might arise if the program accidentally corrupted the variable holding the token code or token map.

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Correct the token code or token map causing the error.

```
-10      ZSPI-ERR-BADSSID    Invalid subsystem ID
```

Cause. A subsystem ID with an invalid name was supplied as a parameter. The owner-name field of a subsystem ID must contain an owner name that:

- Begins with a letter
- Contains only letters, hyphens, or digits
- Is left-justified and padded with blanks

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Check the *ssid* parameter being supplied in the call, and correct it as necessary.

```
-11      ZSPI-ERR-NOTIMP   Operation not supported
```

Cause. This operation is not supported in the version of the SPI definitions being used. (For instance, the operation ZSPI-TKN-ADDR was called to get the address of a header token.)

Effect. The header token ZSPI-TKN-LASTERR is set to this error number, and the requested operation is not completed.

Recovery. Check that the token code being supplied in the call is defined for the version of the SPI definitions being used.

```
-12      ZSPI-ERR-NOSTACK     Insufficient stack space
```

Cause. An SPI procedure was called with fewer words remaining in the data stack than were needed by that procedure.

Effect. The requested operation is not completed.

Recovery. Increase the number of stack pages available or reduce the amount of stack space used. The methods available for taking this action depend on the programming language.

```
-13      ZSPI-ERR-ZFIL-ERR    File-system error
```

Cause. A file-system error occurred.

Effect. The requested operation is not completed.

Recovery. Corrective action depends on the error. Check Table 14-1 for an explanation of the *status-1* and *status-2* parameters.

Table 14-1. Supplementary Status Values

Error	*status-1*	*status-2*	Explanation
-13	2	x	File-system error *x* received during an open of the nonresident template file.
-13	3	x	File-system error *x* received during a read of the nonresident template file.
-14	1	x	ALLOCATESEGMENT error *x* received during allocation of the private segment.
-14	7	x	MOVEX error *x* received during access of the private segment.
-15	-2	x	Nonresident template file error *x*, in which *x* has the following values: -1 File code is not 844. -2 File is not a disk file. -3 File is not key-sequenced. -4 File has wrong record size. -5 File has wrong primary key definition.

```
-14       ZSPI-ERR-ZGRD-ERR    Guardian procedure error
```

Cause. An error occurred during allocation of the private segment; this error is returned from ALLOCATESEGMENT or MOVEX.

Effect. The requested operation is not completed.

Recovery. Corrective action depends on the error. Check Table 14-1 for an explanation of the *status-1* and *status-2* parameters.

```
-15       ZSPI-ERR-INF-FILE    Invalid template file
```

Cause. The template file is invalid.

Effect. The requested operation is not completed.

Recovery. Corrective action depends on the error. Check Table 14-1 for an explanation of the *status-1* and *status-2* parameters.

```
-16       ZSPI-ERR-CONTINUE    More text is available
```

Cause. The entire representation does not fit in the number of lines specified by the *printlines* parameter.

Effect. The amount of text is limited by the *printlines* parameter setting.

Recovery. Call again to get the next portion.

```
-26      ZSPI-ERR-NO-SCANID   See service provider.
```

Cause. No free scan ID was available. Currently, no more than two scan IDs can be active simultaneously.

Effect. The requested operation is not completed.

Recovery. No recovery is possible. Report this error to your service provider.

```
-27      ZSPI-ERR-NO-FORMATID  No format ID available
```

Cause. Currently, no more than one format ID can be active at one time.

Effect. The requested operation is not completed.

Recovery. Recode your program to format only one buffer at a time.

Error Lists

If a HP subsystem calls an SPI procedure and an SPI error occurs—that is, the call returns a nonzero value in `status`—the subsystem normally attempts to recover from the error. If it can't recover (for example, when the application program buffer is corrupted), the subsystem returns information to the application in an error list. The following pages describe the error list associated with each SPI procedure.

The contents of the error list depend on which procedure was called. The standard SPI token ZSPI-TKN-PROC-ERR, which is present in every SPI error list, identifies the procedure.

Each error list always includes the unconditional tokens listed under its description in this subsection. In addition, each error list can include any of the conditional tokens listed under its description.

If you are designing a subsystem that uses SPI, follow these guidelines when constructing an error list for an SPI error:

- Include all unconditional tokens listed in the error-list description.
- Optionally include any or none of the conditional tokens listed in the error-list description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, for additional information about tokens and token types, and for definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

Error From SSGET or SSGETTKN

A call to SSGET or SSGETTKN returned an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
   ZSPI-TKN-PARM-ERR        token-type ZSPI-TYP-PARM-ERR.

ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZSPI-TKN-SSID-ERR        token-type ZSPI-TYP-SSID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the SPI subsystem identifier ZSPI-VAL-SSID. Z-ERROR is the status value from the SPI procedure SSGET or SSGETTKN.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZSPI-VAL-SSGET (2) or ZSPI-VAL-SSGETTKN (3).

ZSPI-TKN-PARM-ERR contains fields ZTOKENCODE, ZINDEX, and ZOFFSET. The value of ZTOKENCODE is the token code or the first 32 bits of the token map passed in the call that failed. ZINDEX is the *index* parameter passed in the call that failed. ZOFFSET is 0.

Conditional Tokens

ZSPI-TKN-SSID-ERR is the *ssid* parameter passed in the call that failed. This token appears only if *ssid* was omitted in the call that failed.

Effect

The call to SSGET or SSGETTKN fails.

Recovery

Follow the recovery procedure for the returned SPI error code as described earlier in this section.

Error From SSINIT

A call to SSINIT returned an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the SPI subsystem identifier ZSPI-VAL-SSID. Z-ERROR is the status value from the SPI procedure SSINIT.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZSPI-VAL-SSINIT (1).

Effect

The call to SSINIT fails.

Recovery

Follow the recovery procedure for the returned SPI error code as described earlier in this section.

Error From SSMOVE or SSMOVETKN

A call to SSMOVE or SSMOVETKN returned an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
   ZSPI-TKN-PARM-ERR        token-type ZSPI-TYP-PARM-ERR.

ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZSPI-TKN-SSID-ERR        token-type ZSPI-TYP-SSID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the SPI subsystem identifier ZSPI-VAL-SSID. Z-ERROR is the status value from the SPI procedure SSMOVE or SSMOVETKN.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZSPI-VAL-SSMOVE (4) or ZSPI-VAL-SSMOVETKN (5).

ZSPI-TKN-PARM-ERR contains fields ZTOKENCODE, ZINDEX, and ZOFFSET. The value of ZTOKENCODE is the token code or the first 32 bits of the token map passed in the call that failed. ZINDEX is the *source-index* parameter passed in the call that failed; ZINDEX is 0 if *source-index* was omitted in the call that failed. ZOFFSET is 0.

Conditional Tokens

ZSPI-TKN-SSID-ERR is the *ssid* parameter passed in the call that failed. This token appears only if *ssid* was omitted in the call that failed.

Effect

The call to SSMOVE or SSMOVETKN fails.

Recovery

Follow the recovery procedure for the returned SPI error code as described earlier in this section.

Error From SSNULL

A call to SSNULL returned an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR        token-type ZSPI-TYP-ENUM.
   ZSPI-TKN-PARM-ERR        token-type ZSPI-TYP-PARM-ERR.

ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the SPI subsystem identifier ZSPI-VAL-SSID. Z-ERROR is the status value from the SPI procedure SSNULL.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZSPI-VAL-SSNULL (6).

ZSPI-TKN-PARM-ERR contains fields ZTOKENCODE, ZINDEX, and ZOFFSET. The value of ZTOKENCODE is the first 32 bits of the token map passed in the call that failed. ZINDEX and ZOFFSET are 0.

Effect

The call to SSNULL fails.

Recovery

Follow the recovery procedure for the returned SPI error code as described earlier in this section.

Error From SSPUT or SSPUTTKN

A call to SSPUT or SSPUTTKN returned an error.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST             token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR         token-type ZSPI-TYP-ENUM.
   ZSPI-TKN-PARM-ERR         token-type ZSPI-TYP-PARM-ERR.
ZSPI-TKN-ENDLIST             token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZSPI-TKN-SSID-ERR         token-type ZSPI-TYP-SSID.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the SPI subsystem identifier ZSPI-VAL-SSID. Z-ERROR is the status value from the SPI procedure SSPUT or SSPUTTKN.

ZSPI-TKN-PROC-ERR is the procedure in which the error occurred. Its value is ZSPI-VAL-SSPUT (7) or ZSPI-VAL-SSPUTTKN (8).

ZSPI-TKN-PARM-ERR contains fields ZTOKENCODE, ZINDEX, and ZOFFSET. The value of ZTOKENCODE is the token code or first 32 bits of the token map passed in the call that failed. ZINDEX and ZOFFSET are 0.

Conditional Tokens

ZSPI-TKN-SSID-ERR is the *ssid* parameter passed in the call that failed. This token only appears if *ssid* was omitted in the call that failed.

Effect

The call to SSPUT or SSPUTTKN fails.

Recovery

Follow the recovery procedure for the returned SPI error code as described earlier in this section.

15 EDITREAD and EDITREADINIT Errors

This section lists and describes the errors returned in the *status* parameter of EDITREAD and EDITREADINIT procedure calls.

Error Codes

This subsection lists each EDITREAD and EDITREADINIT procedure error code and provides a description of each code.

```
0 or greater           NO ERROR
```

Cause. The call was completed successfully. For EDITREAD, a positive value represents the number of characters that the procedure read in the text line.

Effect. None.

Recovery. No corrective action is required.

```
 -1 END-OF-FILE ENCOUNTERED
```

Cause. The procedure encountered an end-of-file in the EDIT file.

- For EDITREADINIT, this means the file was empty.
- For EDITREAD, this condition might indicate that the reading of the file is finished.

Effect. The procedure sets the error code and returns without performing the requested function.

Recovery. Corrective action is application-dependent.

```
 -2 I/O ERROR
```

Cause. A file-system error occurred.

Effect. The procedure sets the error code and returns without performing the requested function.

Recovery. Call FILEINFO to obtain the file-system error, and refer to Section 2, File-System Errors, for corrective action.

```
  -3 TEXT FILE FORMAT ERROR
```

Cause. EDITREADINIT returns this error if any of the following is true:

- The file was not an EDIT file.
- The buffer length was not a power of two in the range 64 through 2048.
- The directory claims the file contains a line with a sequence number greater than 99999.999.
- The file is internally inconsistent.

For EDITREAD, an error exists in the internal format of the file.

Effect. The procedure sets the error code and returns without performing the requested function.

Recovery. For recovery suggestions for files formatted as EDIT files, see the *EDIT User's Guide and Reference Manual*.

```
  -4 SEQUENCE NUMBER ERROR
```

Cause. EDITREAD returns this error if the sequence number of the line about to be read is one of the following:

- Less than that of its predecessor.
- Greater than 99999.999.
- Greater than the largest sequence number in the directory entry for the file page containing the line about to be read.

Effect. The procedure sets the error code and returns without performing the requested function.

Recovery. For recovery suggestions for files formatted as EDIT files, see the *EDIT User's Guide and Reference Manual*.

```
 -5 CHECKSUM ERROR
```

Cause. The edit control block was invalid. There are two possible causes:

- The program called EDITREAD before calling EDITREADINIT.
- The program modified the edit control block, which invalidated the control block.

Note. If you accidentally set the edit control block's reposition bit to 1, you will not receive this error. When the reposition bit is 1, checksum validation is disabled during the call to EDITREAD.

Effect. The operation terminates.

Recovery. Correct the program error that caused the problem.

16 IOEdit Errors

IOEdit allows the run-time libraries of the compilers supported on the operating system to read and write EDIT format files. An application process can use the following IOEdit procedures to access files:

BACKSPACEEDIT	INITIALIZEEDIT
CLOSEALLEDIT	NUMBEREDIT
CLOSEEDIT	OPENEDIT
CLOSEEDIT_	OPENEDIT_
COMPLETEIOEDIT	PACKEDIT
COMPRESSEDIT	POSITIONEDIT
DELETEEDIT	READEDIT
EXTENDEDIT	READEDITP
GETINCREMENTEDIT	UNPACKEDIT
GETPOSITIONEDIT	WRITEEDIT
INCREMENTEDIT	WRITEEDITP

Like the sequential I/O (SIO) procedures, IOEdit is capable of reading and writing EDIT files. EDIT files have a file code of 101.

Most error messages returned by IOEdit are the same as the file-system error codes described in Section 2, File-System Errors. Some have special meaning to IOEdit. The messages with special meanings are further explained here.

A status or code of 0 indicates that no error occurred.

Error Message Forms

IOEdit returns errors in three different forms:

- Errors returned as positive integers, using the same codes as the file-system errors described in Section 2, File-System Errors. These errors can assume a somewhat different meaning when returned by IOEdit, as described below.
- Errors returned as negative integers. These errors are unique to IOEdit and can occur only from operations on EDIT files.
- Errors that result in the immediate abnormal termination of the calling process. No recovery is possible. IOEdit does not return to the caller but instead writes a message to the home terminal and calls ABEND.

The classes of IOEdit errors are discussed in the following subsections.

File-System Error Codes

Errors reported using positive integers are generally the same as those returned by the file system. The positive IOEdit error codes that do not have the same meaning as the file-system errors described in Section 2, File-System Errors, are presented here.

```
2       (%2)      OPERATION NOT ALLOWED ON THIS TYPE OF FILE
```

Cause. IOEdit returns this error when a file is specified that is not an EDIT file (unstructured, file code 101) or is not on a disk device.

Effect. IOEdit cannot operate on this file.

Recovery. Correct the request.

```
10      (%12)     FILE/RECORD ALREADY EXISTS
```

Cause. This error is returned when the same file has already been processed by a previous call. IOEdit does not support multiple concurrent opens of the same file.

IOEdit returns this error when trying to write a record having the same EDIT line number as a record already in the file. IOEdit has no procedure corresponding to WRITEUPDATE; you can rewrite a record only by first deleting it (by calling DELETEEDIT).

Effect. No action is taken.

Recovery. Correct the request.

```
11      (%13)     FILE NAME NOT IN DIRECTORY OR RECORD NOT
                  IN FILE, OR THE SPECIFIED TAPE FILE IS NOT
                  PRESENT ON A LABELED TAPE
```

Cause. IOEdit returns this error when the file does not exist and read only usage is specified. Unlike FILE_OPEN_, OPENEDIT_ creates the file if read-write or write only use is specified and does not return error 11 in that case.

Effect. The requested file was not found.

Recovery. Correct the request.

```
16      (%20)     FILE NUMBER HAS NOT BEEN OPENED
```

Recovery. Most IOEdit procedures return this error when called with the file number of a file that has not been processed by OPENEDIT_.

Effect. The file was not open, so it could not be processed.

Recovery. Use OPENEDIT or OPENEDIT_ to open the file.

```
31      (%37)     UNABLE TO OBTAIN FILE SYSTEM BUFFER SPACE
33      (%41)     I/O PROCESS IS UNABLE TO OBTAIN BUFFER
SPACE
34      (%42)     UNABLE TO OBTAIN FILE SYSTEM CONTROL BLOCK
```

Cause. IOEdit returns these errors when failures occur in the management of space within the EDIT file segment (EFS). The EFS is analogous to the process file segment (PFS) used by the file system.

These errors occur only when the caller has more EDIT files open than the limit specified to INITIALIZEEDIT (the default limit is 30), or when IOEdit needs to enlarge the EFS but its backing disk is too full to allocate another extent.

Effect. Space is not available to continue processing.

Recovery. Close some EDIT files and try again.

```
45      (%55)     FILE IS FULL
```

Cause. The current file size is too small. WRITEEDIT returns this error when the capacity of the file (set by its extent size and maximum number of extents established when the file was created) is about to be exceeded. Unlike the file system, IOEdit allows you to recover from this error by calling EXTENDEDIT to increase the file's capacity to a maximum of 128 megabytes, and then repeating the call to WRITEEDIT that failed.

Effect. IOEdit cannot continue.

Recovery. Call EXTENDEDIT to increase the file's capacity, then try the call to WRITEEDIT again.

```
46      (%56)     INVALID KEY SPECIFIED
```

Cause. Several IOEdit procedures return this error when the EDIT line number specified in the call is less than -3 or greater than 99999999.

Effect. The request is ignored.

Recovery. Specify a valid EDIT line number.

```
59      (%73)     FILE IS BAD
```

Recovery. Most IOEdit procedures return this error when an automatic attempt to recover from a warning condition (previously reported by one of the negative error codes described later in this section) does not succeed. All subsequent calls (except to CLOSEEDIT) for the same file return error 59.

Effect. The file cannot be used.

Recovery. Recovery is not possible.

Error Codes Unique to IOEdit

IOEdit can detect some error conditions for which no existing file-system error code has been defined. These conditions can occur only in operations on EDIT files. IOEdit uses negative integers to report these conditions.

Error numbers -1 through -5 are returned when IOEdit finds something wrong with the directory in an EDIT file. This directory corresponds to the set of index blocks in a key-sequenced file and can be rebuilt by reading the entire file. Rather than doing this, OPENEDIT_ sets a flag and returns one of the error codes -6 through -10.

The caller can then choose to abandon the file by calling CLOSEEDIT or try to reconstruct the file's directory by calling any of the other IOEdit procedures. In the latter case, IOEdit tries to rebuild the directory in memory, without changing the file on the disk. If this succeeds, all subsequent IOEdit operations work as if the file were intact. If this does not succeed, this and all subsequent calls to IOEdit procedures (except CLOSEEDIT) for the same file return error 59.

```
-1   (inconsistent directory size value)
```

Cause. A directory error occurred in an EDIT file.

Effect. Processing stops.

Recovery. Proceed as described at the beginning of this subsection.

```
-2   (record numbers in the directory are out of order)
```

Cause. A directory error occurred in an EDIT file.

Effect. Processing stops.

Recovery. Proceed as described at the beginning of this subsection.

```
-3   (a record number in the directory is beyond legal range)
```

Cause. A directory error occurred in an EDIT file.

Effect. Processing stops.

Recovery. Proceed as described at the beginning of this subsection.

```
-4   (a block number in the directory is outside the file)
```

Cause. A directory error occurred in an EDIT file.

Effect. Processing stops.

Recovery. Proceed as described at the beginning of this subsection.

```
-5   (duplicate block numbers in the directory)
```

Cause. A directory error occurred in an EDIT file.

Effect. Processing stops.

Recovery. Proceed as described at the beginning of this subsection.

```
-6   (numbers ran out)
```

Cause. COMPRESSEDIT and EXTENDEDIT return this error when the new record number of the last record in the file would be larger than 99999999. NUMBEREDIT returns this error when the new record number of the last record to be renumbered would not be less than the record number of the next record.

Effect. In either case, the problem is that the increment parameter's value is too large, and the procedure does nothing.

Recovery. Change the increment parameter to an acceptable value.

```
-7   (record numbers out of order)
```

Cause. DELETEEDIT and NUMBEREDIT return this error when the record number of the first record to be deleted or renumbered is greater than that of the last.

Several other IOEdit procedures return this error when reading a file and finding a record whose record number is not greater than that of the record preceding it.

Effect. The procedure stops.

Recovery. The file is bad but possibly most of the file's contents could be recovered by copying the entire file and ignoring only the records having this error.

```
-8   (record length too large)
```

Cause. Several IOEdit procedures return this error when reading a file and finding a record whose length is such that the record would extend beyond the end of its block.

Effect. The procedure stops.

Recovery. The file is bad but possibly most of the file's contents could be recovered by copying the entire file and ignoring only the records having this error.

```
-9   (record format inconsistent)
```

Cause. Several IOEdit procedures return this error when reading a file and finding a record whose compression codes disagree with the length of the record.

Effect. The procedure stops.

Recovery. The file is bad but possibly most of the file's contents could be recovered by copying the entire file and ignoring only the records having this error.

```
-10   (cannot do renumbering)
```

Cause. NUMBEREDIT returns this error when the parameters represent disallowed renumbering: the first new record number is not greater than that of the record preceding the first record to be renumbered, the new record number increment is zero or negative, or no records are affected.

Effect. The procedure stops.

Recovery. Recovery is not possible.

```
-13   (error accessing non-EDIT file line numbers)
-14   (error accessing non-EDIT file change tags)
-15   (change tag should be appended to record)
-16   (both tag and record number error)
-17   (change tag overlays text)
```

Cause. These errors can occur only when IOEdit is accessing a file other than an EDIT file. For example, they can occur when a local version control system database is accessed.

Effect. The procedure stops.

Recovery. This error should not occur; recovery is not possible. Determine the type of access desired and correct the request.

Catastrophic Error Messages

Errors for which no recovery is possible result in the immediate abnormal termination of the calling process. For these conditions, IOEdit does not return to the caller but instead writes a message to the home terminal and calls ABEND. The catastrophic error messages are the following.

```
**** IOEdit internal error, process aborted ****
```

Cause. A data structure maintained by IOEdit has an inconsistent condition. The problem is in IOEdit itself, not in the calling program.

Effect. The calling process is terminated abnormally.

Recovery. Recovery is not possible.

```
**** IOEdit unable to allocate Edit File Segment, process
     aborted ****
```

Cause. If the caller does not explicitly call the INITIALIZEEDIT procedure, the first IOEdit procedure called does so automatically. In any case, INITIALIZEEDIT first tries to create the Edit File Segment on the swapping disk volume for the calling process, and if that fails, it then tries every disk volume in the system. If all attempts fail, IOEdit issues this message and calls ABEND. If the system is that short of disk space, many other processes will probably fail in a similar manner.

Effect. The calling process is terminated abnormally.

Recovery. Recovery is not possible.

```
**** IOEdit stack overflow, process aborted ****
```

Cause. In the course of accessing caller-provided information that is in user extended data segments, IOEdit makes local copies on the stack. This can make the program's stack space requirements much larger than was apparent when the program was compiled and bound. When the stack is about to overflow due to such an action, IOEdit issues this message rather than allowing a mysterious trap to occur. The user should rearrange global primary, global secondary, and the stack space use of the calling procedures leading to the call to IOEdit to increase the space available for the stack within IOEdit.

Effect. The calling process is terminated abnormally.

Recovery. Recovery is not possible.

```
**** IOEdit needs more pages in user data segment, process
     aborted ****
```

Cause. A program using IOEdit might require more data pages than was apparent when the program was compiled and bound. When IOEdit makes a local copy on the stack and these additional pages would cause an address trap because the process has fewer than 32 data pages (as indicated by the value returned by LASTADDR), IOEdit issues this message rather than allowing a mysterious trap to occur. The user should rerun the program with MEM 32 (or more) specified in the RUN command.

Effect. The calling process is terminated abnormally.

Recovery. Recovery is not possible.

17 Formatter Errors

The following errors are returned by the FORMATDATA and FORMATDATAX procedures. The error code is returned as the resulting value from the procedure call.

For additional information about the FORMATDATA[X] procedures, refer to the *Guardian Programmer's Guide*.

Error Codes

This subsection lists each FORMATDATA[X] procedure error code and provides a description of each code.

```
267  BUFFER OVERFLOW
```

Cause. To interpret an edit descriptor, FORMATDATA[X] had to access a character before the start of the buffer or a character outside the buffer.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the format.

```
268  NO BUFFER
```

Cause. FORMATDATA[X] required a new buffer, but there were no more buffers.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the format, or increase the number of buffers.

```
270  FORMAT LOOPBACK
```

Cause. Data items remained to be processed after FORMATDATA[X] reached the end of a format that contains no repeatable edit descriptors.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Include repeatable edit descriptors in the format, or reduce the number of data items.

```
271  EDIT ITEM MISMATCH
```

Cause. In a format-directed operation, an edit descriptor was matched to a data element that has an incompatible type. For example, the "G" edit descriptor was associated with a string data element on output, or any edit descriptor except "A" was associated with a string data element on input.

In list-directed input, a numeric data element was repeated using the r*c form, and some data element after the first element to which this form applied was a string-type element.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the format, or correct the data list to include missing (or delete extra) items.

```
272  ILLEGAL INPUT CHARACTER
```

Cause. The numeric input field contained an inappropriate character for the corresponding edit descriptor. For example, a nonnumeric character was entered in a field being interpreted according to the "I" edit descriptor, or lowercase letters were used where uppercase letters were required.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the format or the data list.

```
273  BAD FORMAT
```

Cause. The format contains an edit descriptor that is valid for output but not for input. For example, I5.5 is invalid for input.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Correct the format.

```
274  NUMERIC OVERFLOW
```

Cause. A data element's numeric value was so small or so large that FORMATDATA[X] could not place it in its corresponding data element.

Effect. The procedure sets the error code and returns without performing the requested operation.

Recovery. Change the format or correct the numeric calculations.

18 INITIALIZER Errors

The following error messages are returned by the INITIALIZER procedure. No error numbers are returned from this procedure call.

For additional information about the INITIALIZER procedure, refer to the *Guardian Programmer's Guide* and the *Guardian Procedure Calls Reference Manual*.

The following messages indicate fatal coding or internal errors and are issued only when the program calls the ABEND procedure. The error message is passed along in the ABEND procedure call to the process's creator.   If the creator process is the TACL process, the message is passed to the home terminal. If the creator process is not the TACL process, error handling is application dependent.

Error Messages

This subsection lists each INITIALIZER procedure error message and provides a description of each message.

```
Error message: "INITIALIZER: Unable to allocate buffer for
                message"
```

Cause. An error occurred during the allocation of space in a buffer pool. If the procedure was called from a FORTRAN or COBOL program, the SAVE directive was not specified.

Effect. The INITIALIZER prepares the error message and calls ABEND.

Recovery. If the procedure was called from a TAL program, this is an internal error. If bits <0:10> of the *flags* parameter are 0, report this problem to your service provider.

If the procedure was called from a FORTRAN or COBOL program, specify the MEM 64 option in your RUN command or reduce the number of ASSIGN and PARAM attributes for your program.

```
Error message: “INITIALIZER: Invalid FCB format or wrong
               number of FCBs specified”
```

Cause. The cause of this error can be one of the following:

- The file control block (FCB) size is not correct.
- An invalid number of FCBs were specified in either the ALLOCATE^CBS or ALLOCATE^CBS^D00 DEFINE.

Effect. The INITIALIZER prepares the error message and calls ABEND.

Recovery. Correct the FCB size. The correct sizes for pTAL or TAL programs are as follows:

DEFINE Used	FCB Size for TNS/R Native Process	FCB Size for TNS or Accelerated Process
ALLOCATE^CBS	82 words	60 words
ALLOCATE^CBS^D00	102 words	80 words

If the FCB size was correct, check the number of FCBs in the ALLOCATE^CBS or ALLOCATE^CBS^D00 DEFINE.

```
Error message: “INITIALIZER: The NUM^FCBS parameter is
               incorrect or not specified”
```

Cause. The cause of this error can be one of the following:

- The INITIALIZER procedure was called by a TNS/R native process that did not specify the required *num^fcbs* parameter.
- A negative number of file control blocks (FCBs) was specified in the *num^fcbs* parameter.
- A positive number of file control blocks (FCBs) was specified in the *num^fcbs* parameter but the *fcb^array* parameter was not specified.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. Either correct the value in the *num^fcbs* parameter or specify the *num^fcbs* parameter.

```
Error message: “INITIALIZER: Timeout reading $RECEIVE”
```

Cause. A timeout occurred during a wait on $RECEIVE to read the startup sequence.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. The default waiting period is 60 seconds. The D10 and later versions of GPLIB offer an optional parameter to specify this timeout period.

```
Error message: “INITIALIZER: Unexpected message from
               creator process”
```

Cause. The message received from the creator process was not a valid message in the startup sequence.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. Correct the program that is sending the startup sequence. Refer to the *Guardian Programmer’s Guide* for more information.

```
Error message: “INITIALIZER: Message saving requested but
               RUCB missing or undersized”
```

Cause. This is an internal error.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. If bits <0:10> of the *flags* parameter are 0, report this problem to your service provider.

```
Error message: “INITIALIZER: Unable to obtain process handle”
```

Cause. This is an internal error.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. Report this problem to your service provider.

```
Error message: “INITIALIZER: Backup takeover and
               flags.<12> reset”
```

Cause. The backup of a process pair returned from CHECKMONITOR and *flags*.<12> was reset to 0.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. Either correct the problem that caused the primary process to terminate, make sure the primary process does stack checkpointing, or set bit 12 of the *flags* parameter to 1.

```
Error message: “INITIALIZER: Unable to open $RECEIVE”
```

Cause. $RECEIVE is probably already open.

Effect. INITIALIZER prepares the error message calls ABEND.

Recovery. Make sure $RECEIVE is not open when you call INITIALIZER.

```
Error message: “INITIALIZER: Unexpected error”
```

Cause. This is an internal error.

Effect. INITIALIZER prepares the error message and calls ABEND.

Recovery. Report this problem to your service provider.

19 Interprocess Command Interpreter Messages

Application processes use Tandem Advanced Command Language (TACL) to send or receive the messages described in this section. The TACL product is the command interpreter supplied by HP for use on the operating system.

Though user-written command interpreters can be used, in this section the TACL product is assumed to be the only command interpreter. If a user-written command interpreter is in use, read "command interpreter" instead of "TACL" in the following messages.

The interprocess command interpreter messages described in this section should not be confused with the system messages described in Section 20, System Messages. Though these messages share many of the same message numbers, the command interpreter messages do not generate a file-system error 6 (system message received) as do the system messages.

Command interpreter messages are not error messages; they are used to convey information. When a request for a new process occurs, such as from a PROCESS_CREATE_ procedure call, the operating system creates a new process. When the process is created, a message is sent to it by the TACL process. This message is called the startup message. The startup message carries an indication of any ASSIGN or PARAM messages to be added. (ASSIGNs and PARAMs establish certain characteristics of the new process.) When there are no more ASSIGNs or PARAMs to be processed, the newly created process is ready to be accessed.

Figure 19-1 illustrates the communication between an application process, the TACL process, and $CMON.

Figure 19-1. Command Interpreter Messages

```
application    <---->       TACL       <---->       $CMON
```

Messages -1, -2, and -3 are sent by the TACL process to processes that it creates. Messages -20 and -21 are sent by any process to the TACL process.

Messages -50 through -60 are sent to $CMON, but not all of these messages are sent by the TACL process. ADDUSER, DELUSER, PASSWORD, and RPASSWORD are privileged, licensed programs; they send their messages to $CMON. When $CMON is running, these programs add an additional level of access restriction to the operating system. When $CMON is not running, their restrictions are not in effect.

The first word of each $CMON message is usually a control word that permits or disallows an operation. For most $CMON messages, the rest of the reply is text that the TACL process displays. It typically contains the reason why $CMON did not permit the operation. $CMON messages are also used to display the greeting at LOGON or a farewell message at LOGOFF. Two exceptions are the command interpreter messages -52 (run) and -60 (configuration); these allow $CMON to return information or to display text. If the first word is zero, the rest of the reply contains data; if the first word is nonzero, the rest of the reply contains text to be displayed.

The sources of the interprocess command interpreter messages are shown in Table 19-1.

Table 19-1. Command Interpreter Messages

CI Message	Source
-1 Startup	Sent by TACL process to the process it created
-2 Assign	Sent by TACL process to the process it created
-3 Param	Sent by TACL process to the process it created
-20 Wakeup	Sent by any process to TACL process
-21 Display	Sent by any process to TACL process
-50 Logon	Sent to $CMON by TACL process
-51 Logoff	Sent to $CMON by TACL process
-52 Run	Sent to $CMON by TACL process
-53 Illegal Logon	Sent to $CMON by TACL process
-54 Add User	Sent to $CMON in adduser^msg
-55 Delete User	Sent to $CMON in deluser^msg
-56 Alter Priority	Sent to $CMON by TACL process
-57 Password	Sent to $CMON in password^msg
-58 Remote Password	Sent to $CMON in remotepassword^msg
-59 Prelogon	Sent to $CMON by TACL process
-60 Configuration	Sent to $CMON by TACL process

The message lengths in this section are subject to change. Do not test the received length against an expected length.

For more information about command interpreter messages, refer to the *Guardian Programmer's Guide*.

Message Descriptions

This subsection describes the interprocess command interpreter messages.

```
  -1  STARTUP
```

Cause. This message is received by a new process; it indicates that the new process was successfully created.

Format. The Startup message always ends with a null byte to terminate the parameter string. If the resulting message has an odd number of bytes, the TACL process appends a second null byte. The maximum length possible for a startup message is 596 bytes (including the trailing null characters).

The form of the startup message is:

```
STRUCT ci^startup;
  BEGIN                  !  word
  INT msgcode;           !  [0] -1

STRUCT default;
  BEGIN
  INT volume [0:3];      !  [1] $default-volume-name
  INT subvol [0:3];      !      default-subvol-name
  END;

STRUCT infile;
  BEGIN
  INT volume [0:3];      !  [9] IN parameter file name
  INT subvol [0:3];      !      of the RUN command
  INT dname  [0:3];
  END;

STRUCT outfile;
  BEGIN
  INT volume [0:3];      ! [21] OUT parameter file name
  INT subvol [0:3];      !      of the RUN command
  INT dname  [0:3];
  END;

  STRING param [0:n-1];  ! [33] parameter string of the RUN
                         !      command (if any) that was
                         !      entered by the operator.
                         !      This is in either of the
                         !      following forms:
                         !
                         !   parameter-string null [null]
                         !     or
                         !   null  null

                         !   n = ( count-read - 66 )
  END;
```

Response. If you want your program to receive any existing ASSIGN and PARAM messages, specify file-system error code 70 in a call to REPLY or specify error code 0 but with a reply of 1 to 4 bytes, where bit 0 of the first byte is set to 1 for ASSIGN messages and bit 1 is set to 1 for PARAM messages.

```
-2  ASSIGN
```

Cause. A new process receives one ASSIGN message for each assignment in effect at the time the new process was created.

The ASSIGN messages immediately follow the startup message if the new process does one of the following:

- Replies to the startup message with an error return value of 70. The TACL process then sends both ASSIGN and PARAM messages.
- Replies to the startup message with an error return value of 0 but with a reply of from 1 through 4 bytes, where bit 0 of the first byte of the reply is set to 1. If bit 1 of the first byte of the reply is set to 1, the TACL process also sends a PARAM message.

Format. The format of the ASSIGN message follows. The message length is 108 bytes.

```
STRUCT ci^assign;           ! ASSIGN message
  BEGIN                     !
  INT msg^code;             ! [0] -2
                            !
STRUCT logicalunit;         ! PARAMETERS TO ASSIGN COMMAND
  BEGIN                     !
  STRING prognamelen;       ! [1] length in bytes of name
  STRING progname[0:30];    !      {0:31} {program-unit | *}
                            !      (blank filled on right)
  STRING filenamelen;       ! [17] length in bytes of name
  STRING filename[0:30];    !      {0:31} logical-file-name
                            !      (blank filled on right)
  END;                      !

  INT(32) fieldmask;        ! [33] bit mask to indicate
                            !      which of the following
                            !      fields were supplied
                            !      (1 = supplied):
                            !
                            !      .<0> = physical-filename
                            !      .<1> = pri-extent-size
                            !      .<2> = sec-extent-size
                            !      .<3> = file-code
                            !      .<4> = exclusion-spec
                            !      .<5> = access-spec
                            !      .<6> = record-size
                            !      .<7> = block-size
```

```
STRUCT physicalfilename;  ! [35] physical-filename
  BEGIN                   !
  INT volume [0:3];       !
  INT subvol [0:3];       !
  INT dfile  [0:3];       !
  END;                    !
                          !
  INT primaryextent;      ! [47] pri-extent-size
  INT secondaryextent;    ! [48] sec-extent-size
  INT filecode;           ! [49] file-code
  INT exclusionspec;      ! [50] %00 if SHARED    }
                          !      %20 if EXCLUSIVE } flag
                          !      %60 if PROTECTED } param
  INT accessspec;         ! [51] %0000 if I/O     } of
                          !      %2000 if INPUT   } OPEN
                          !      %4000 if OUTPUT  }
  INT recordsize;         ! [52] record-size
  INT blocksize;          ! [53] block-size
  END;
```

Response. The application need not respond to the ASSIGN message. When all of the ASSIGN and PARAM messages have been received, the process is ready for use.

```
-3  PARAM
```

Cause. A new process receives a PARAM message if any parameters are in effect when the new process is created.

The PARAM messages immediately follow the ASSIGN messages if the process does one of the following:

- Replies to the startup message with an error return value of 70. The TACL process then sends both ASSIGN and PARAM messages.
- Replies to the startup message with an error return value of 0, but with a reply of from 1 through 4 bytes, where bit 1 of the first byte of the reply is set to 1. If bit 0 of the first byte of the reply is set to 1, the TACL process also sends ASSIGN messages.

Format. The format of the PARAM message follows. The maximum message length is 1028 bytes.

```
STRUCT ci^param;                ! PARAM message
  BEGIN                         !
  INT msg^code;                 ! [0] -3
  INT numparams;                ! [1] number of
                                !     parameters
                                !     included in
                                !     this message
  STRING parameters [0:1023];   ! [2] beginning of
                                !     parameters
  END;
```

The field *parameters* in the above message format is composed of *numparams* records of the following form (offsets are given in bytes):

```
param[0]         = length n, in bytes, of
                     parameter-name
param[1] FOR n   = parameter-name
param[n+1]       = length v, in bytes, of
                     parameter-value
param[n+2] FOR v = parameter-value
```

Response. The application need not respond to the ASSIGN message. When all of the ASSIGN and PARAM messages have been received, the process is ready for use.

```
-20  WAKEUP
```

Cause. A process sent a wakeup message to the TACL process.

If the TACL process is paused, the wakeup message causes it to return to the command input mode (that is, "wake up"). If the TACL process is not paused, it ignores the wakeup message.

Format. The format of the wakeup message follows. The message length is 2 bytes.

```
STRUCT wakeup^msg;
   BEGIN
   INT msgcode;         ! -20
   END;
```

Response. None required.

```
-21  DISPLAY
```

Cause. A process sends a display message to the TACL process.

The display message causes the TACL process to display the text contained in the message. The text is displayed just prior to the next time the TACL process prompts for a command.

The TACL process can store 8 undisplayed, 132-byte messages. If 8 messages are stored, subsequent messages are rejected with an error 12 indication (file in use).

If the TACL process receives a user message followed by a zero-length message, the TACL process clears its $RECEIVE buffer and checks the message length.

Format. The format of the display message follows. The message length is 2 bytes plus the display-text length (in bytes).

The length of the text portion is implied in the write count used to send this message.

```
STRUCT display^msg;
  BEGIN
  INT msgcode;                ! -21
  STRING text [0:n-1];        ! n <= 132
  END;
```

Response. None required.

```
 -50  LOGON
```

Cause. This message is sent to the $CMON process every time the TACL process tries to log on. If the $CMON process is not running, no $CMON logon restrictions are in effect.

When a LOGON command is entered, the user name is checked for validity.

Format. The format of the logon message follows. The message length is 54 bytes.

```
STRUCT logon^msg;
  BEGIN
  INT msgcode;              !  [0] -50
  INT userid;               !  [1] user ID of user
                            !      logging on
  INT cipri;                !  [2] initial execution
                            !      priority of TACL
  INT ciinfile  [0:11];     !  [3] name of the TACL
                            !      command file
  INT cioutfile [0:11];     ! [15] name of the TACL
                            !      list file
  END;
```

Response. The $CMON reply indicates whether the user is allowed to log on and contains an optional display message in the following form:

```
STRUCT logon^reply;
  BEGIN
  INT replycode;            !  [0] 0 = allow logon
                            !      1 = disallow logon
  STRING                    !
      replytext [0:n];      !  [1] optional message to
                            !      be printed; maximum
  END;                      !      of 132 bytes
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
 -51  LOGOFF
```

Cause. This message is sent to the $CMON process when a LOGOFF command is entered. It is also sent when a user logs on without first logging off (implicit logoff).

Format. The form of the logoff message follows. The message length is 54 bytes.

```
STRUCT logoff^msg;
  BEGIN
  INT msgcode;               !  [0] -51
  INT userid;                !  [1] user ID of user logging
                             !      off
  INT cipri;                 !  [2] initial execution
                             !      priority of TACL
  INT ciinfile  [0:11];      !  [3] name of the TACL
                             !      command file
  INT cioutfile [0:11];      ! [15] name of the TACL
                             !      list file
  END;
```

Response. The $CMON reply contains an optional display message. If the $CMON process is not running, the TACL process does not try to write the logoff message. The format of the reply to the logoff message is:

```
STRUCT logoff^reply;
  BEGIN
  INT replycode;             !  [0] ignored by TACL
  STRING                     !
      replytext [0:131];     !  [1] optional message to be
                             !      printed; maximum of 132
                             !      bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
-52  RUN
```

Cause. This message is sent to the $CMON process whenever the user tries to start a process either explicitly (RUN *prog-file*), implicitly (*prog-file*), or with the TACL #NEWPROCESS built-in function.

The RUN parameters IN *file*, OUT *file*, LIB *file*, SWAP *file*, and the parameter string are included in the process-creation message sent to $CMON.

Format. The format of the RUN message follows. The length of the message is determined by the value of *paramlen*.

```
STRUCT processcreation^msg;
  BEGIN
  INT msgcode;               !  [0] -52
  INT userid;                !  [1] user ID of user logged on
  INT cipri;                 !  [2] initial priority of TACL
  INT ciinfile    [0:11];    !  [3] name of TACL command file
  INT cioutfile   [0:11];    ! [15] name of the TACL list file
  INT progname    [0:11];    ! [27] expanded program file
                             !      name
  INT priority;              ! [39] the value of the PRI RUN
                             !      parameter if supplied;
                             !      otherwise, -1
  INT processor;             ! [40] the value of the processor RUN
                             !      parameter if supplied;
                             !      otherwise, -1
  INT proginfile  [0:11];    ! [41] the expanded IN file RUN
                             !      parameter if supplied;
                             !      otherwise, the default
                             !      IN file
  INT progoutfile [0:11];    ! [53] the expanded OUT file RUN
                             !      parameter if supplied;
                             !      otherwise, the
                             !      default OUT file
  INT proglibfile [0:11];    ! [65] the expanded LIB file
                             !      RUN parameter if
                             !      supplied; otherwise,
                             !      blanks
  INT progswapfile [0:11];   ! [77] the expanded SWAP file
                             !      RUN parameter if supplied;
                             !      otherwise, blanks
  INT paramlen;              ! [89] the length of param. This is
                             !      defined for D42 and later
                             !      releases of the operating
system
  STRING param [0:527]       ! [90] parameter string of the RUN
                             !      command, which is up to 528
                             !      bytes in length including 2
null
                             !      bytes at the end of the
string.
                             !      This is defined for D42 and
                             !      later releases of the
                             !     operating system.
END;
```

The $CMON process can reply in one of two ways:

- With a run-the-process reply
- With a disallow-process-creation reply

A run-the-process reply contains the process priority and the name of the processor where the process will run. This reply has the format:

```
STRUCT processcreation^reply;
   BEGIN
   INT replycode;           !   [0] 0 = create the process
   INT progname [0:11];     !   [1] expanded name of program
file
                            !       to be run
   INT priority;            !  [13] execution priority of the
                            !       new process.
                            !       0 = PRI option specified by
                            !           user.
                            !           If no PRI option is
                            !           specified, then TACL
                            !           priority minus 1.
                            !      >0 = execution priority
                            !      <0 = PRI option specified by
                            !           user plus negative
                            !           priority offset
returned
                            !           in this field.
                            !           If no PRI option is
                           !           specified by user, then
                            !           TACL priority minus 1
                            !           plus negative priority
                            !           priority offset
returned
                            !           in this field.
                            !           For example, PRI = 150,
                           !           priority = -5, priority
                            !           used = 145).
  INT processor;            !  [14] processor where new
                            !       process is to run or -1.
                            !       If -1, then the processor
                            !       in which the TACL process
                            !       is running is used.
  END;
```

The values returned in the reply are those used for the process-creation attempt. Any process-creation errors are seen by the TACL user (no notification is made to $CMON).

A disallow-process-creation reply has the format:

```
STRUCT processcreation^reply;
  BEGIN
  INT replycode;             !   [0] 1 = disallow process
                             !           creation
  STRING                     !
      replytext [0:131];     !   [1] optional message to be
                             !       printed
  END;
```

Response. The length of the message is 2 bytes plus the *replytext* length in bytes. The length of the *replytext* is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
 -53  ILLEGAL LOGON
```

Cause. A user tried to log on three times in a row and failed each time.

Format. The format of the illegal logon message follows. No byte count is returned.

```
STRUCT illegal^logon^msg;
  BEGIN
  INT msgcode;                !  [0] -53
  INT userid;                 !  [1] user ID of user trying
                              !      to log on
  INT cipri;                  !  [2] initial priority of TACL
  INT ciinfile    [0:11];     !  [3] name of the TACL command
                              !      file
  INT cioutfile   [0:11];     ! [15] name of the TACL list
                              !      file
  STRING
      logonstring [0:n];      ! [27] the attempted logon
                              !      command string;
                              !      maximum length of 132
                              !      bytes
  END;
```

Response. The $CMON reply message contains an optional display message. The format for the reply is:

```
STRUCT illegal^logon^reply;
  BEGIN
  INT replycode;              ! [0] ignored by TACL
  STRING replytext [0:n];     ! [1] optional message to be
                              !     printed; maximum length
                              !     of 132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

The TACL process delays for one minute.

```
 -54  ADD USER
```

Cause. A user tried to add another user to the system. Any user can be added if the current user's security setting allows it. However, if $CMON is not running, no $CMON add-user restrictions are in effect.

Format. The format of the add-user message is:

```
STRUCT adduser^msg;
  BEGIN
  INT msgcode;              !  [0] -54
  INT userid;               !  [1] user ID of user adding the
                            !      new user
  INT cipri;                !  [2] initial priority of TACL
  INT ciinfile  [0:11];     !  [3] name of the TACL
                            !      command file
  INT cioutfile [0:11];     ! [15] name of the TACL
                            !      list file
  INT groupname [0:3];      ! [27] the group name of the user
                            !      being added
  INT username  [0:3];      ! [31] the user name of the user
                            !      being added
  INT group^id;             ! [35] the group number of the
                            !      user being added
  INT user^id;              ! [36] the user number of the
                            !      user being added
  END;
```

Response. The $CMON reply indicates whether the user can be added and contains an optional display message. The format of the reply message is:

```
STRUCT adduser^reply;
  BEGIN
  INT replycode;                ! [0] 0 = allow addition of
                                !         user
                                !     1 = disallow addition
                                !         of user
  STRING replytext [0:n];       ! [1] optional message to be
                                !     printed; maximum length
                                !     of 132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
 -55  DELETE USER
```

Cause. A user tried to delete another user from the system. Any user can be deleted if the current user's security setting allows it. If the $CMON process is not running, then no $CMON delete-user restrictions are in effect.

Format. The format of the delete-user message is:

```
STRUCT deluser^msg;
  BEGIN
  INT msgcode;           !  [0] -55
  INT userid;            !  [1] user ID of user deleting
                         !      user
  INT cipri;             !  [2] initial priority of TACL
  INT ciinfile  [0:11];  !  [3] name of the TACL
                         !      command file
  INT cioutfile [0:11];  ! [15] name of the TACL list file
  INT groupname [0:3];   ! [27] the group name of the user
                         !      being deleted
  INT username  [0:3];   ! [31] the user name of the user
                         !      being deleted
  END;
```

Response. The $CMON reply indicates whether the user should be deleted and contains an optional reply message. The format of the reply message is:

```
STRUCT deluser^reply;
  BEGIN
  INT replycode;          ! [0] 0 = allow deletion of user
                          !     1 = disallow deletion of
                          !         user
  STRING replytext [0:n]; ! [1] optional message to be
                          !     printed; maximum length of
                          !     132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
-56  ALTER PRIORITY
```

Cause. A user tried to alter the priority of a process. A user can change the priority of any process that has the same access ID as that user. Only someone logged on as a super-group user can change the priority of any process. However, if the $CMON process is not running, then no $CMON alter-priority restrictions are in effect.

Format. The format of the alter-priority message is:

```
STRUCT altpri^msg;
  BEGIN
  INT msgcode;           !  [0] -56
  INT userid;            !  [1] user ID of user altering
                         !      the priority
  INT cipri;             !  [2] initial priority of TACL
  INT ciinfile  [0:11];  !  [3] name of the TACL
                         !      command file
  INT cioutfile [0:11];  ! [15] name of the TACL list file
  INT crtpid    [0:3];   ! [27] process ID of the process
                         !      whose priority is to be
                         !      altered
  INT progname  [0:11];  ! [31] expanded program file name
                         !      of the process whose
                         !      priority is to be altered
  INT priority;          ! [43] the new priority
  INT phandle   [0:9];   ! [44] process handle of the process
                         !      whose priority is to be
                         !      altered
  END;
```

Response. The $CMON reply indicates whether the process priority should be changed and contains an optional display message. The format of the reply message is:

```
STRUCT altpri^reply;
  BEGIN
  INT replycode;            ! [0] 0 = allow priority to be
                            !         altered
                            !     1 = disallow priority to
                            !         be altered
  STRING replytext [0:n];   ! [1] optional message to be
                            !     printed; maximum length
                            !     of 132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
 -57  PASSWORD
```

Cause. The user tried to change his or her password. Users can change their passwords at any time. However, if the $CMON process is not running, then no $CMON password restrictions are in effect.

Format. The format of the password message is:

```
STRUCT password^msg;
  BEGIN
  INT msgcode;            !  [0] -57
  INT userid;             !  [1] user ID of the user
                          !      changing the password
  INT cipri;              !  [2] initial priority of TACL
  INT ciinfile  [0:11];   !  [3] name of the TACL
                          !      command file
  INT cioutfile [0:11];   ! [15] name of the TACL list file
  END;
```

Response. The $CMON reply indicates whether the user password can be changed and contains an optional display message. The format of the reply message is:

```
STRUCT password^reply;
  BEGIN
  INT replycode;           ! [0] 0 = allow password to be
                           !         changed
                           !     1 = disallow password to
                           !         be changed
  STRING replytext [0:n];  ! [1] optional message to be
                           !     printed; maximum length of
                           !     132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
-58  REMOTE PASSWORD
```

Cause. The user tried to change his or her remote password. Users can change their remote passwords at any time. However, if the $CMON process is not running, then no $CMON remote-password restrictions are in effect.

Format. The format of the remote password message is:

```
STRUCT remotepassword^msg;
  BEGIN
  INT msgcode;               !  [0] -58
  INT userid;                !  [1] user ID of user changing
                             !      remote password
  INT cipri;                 !  [2] initial priority of TACL
  INT ciinfile  [0:11];      !  [3] name of the TACL
                             !      command file
  INT cioutfile [0:11];      ! [15] name of the TACL list file
  INT sysname [0:3];         ! [27] change the remote password
                             !      for this system
                             !      ("*" indicates all systems)
  END;
```

Response. The $CMON reply indicates whether the user's remote password can be changed and contains an optional display message. The format of the reply message is:

```
STRUCT remotepassword^reply;
  BEGIN
  INT replycode;              ! [0] 0 = allow the remote
                              !         password to be
                              !         changed
                              !     1 = disallow the remote
                              !         password to be
                              !         changed
  STRING replytext [0:n];     ! [1] optional message to be
                              !     printed; maximum length
                              !     of 132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
 -59  PRELOGON
```

Cause. The TACL process tried to log on. If the $CMON process is not running, then no $CMON restrictions are in effect. This message is sent before the TACL process calls VERIFYUSER.

Format. The form of the prelogon message follows. The length of the message is 72 bytes.

```
STRUCT prelogon^msg;
  BEGIN
  INT msgcode;              !  [0] -59
  INT userid;               !  [1] user ID of user logging on:
                            !      0 if the user is logged off
                            !      or is logged on as
                            !      NULL.NULL (0,0)
  INT cipri;                !  [2] current priority of TACL
  INT ciinfile  [0:11];     !  [3] TACL IN file
  INT cioutfile [0:11];     ! [15] TACL OUT file
  INT loggedon;             ! [27] 0 if TACL is currently
                            !      logged off, non-zero if
                            !      TACL is already logged on
  INT username [0:7];       ! [28] internal username through
                            !      which the user wants to
                            !      log on
  END;
```

Response. The $CMON reply indicates whether the user can log on and contains an optional display message. The form of the reply message is:

```
STRUCT prelogon^reply;
  BEGIN
  INT replycode;             ! [0] 0 = proceed to VERIFYUSER
                             !     1 = disallow logon
  STRING replytext [0:n];    ! [1] optional message to be
                             !     printed; maximum length
                             !     of 132 bytes
  END;
```

The length of the message is 2 bytes plus the reply-text length in bytes. The length of the reply text is implied in the reply count used when making a reply. If *reply-count* = 2, no text is displayed.

```
 -60  CONFIGURATION
```

Cause. This message is sent to the $CMON process just before either an interactive TACL process attempts to log on from the logged-off state or a noninteractive TACL process starts. You can set `config^request^typ`e to 1 to obtain configuration data after log on. If `requestcmonuserconfig` is set to 1 in the reply message, TACL should request $CMON for user configuration data after log on. For more information on writing a command interpreter, refer to the *Guardian Programmer's Guide*.

If the $CMON process is not running or is running too slowly, the TACL configuration remains unchanged from its previous values.

Format. The format of the configuration message is:

```
STRUCT config^msg;
  BEGIN

  INT msgcode;              ! [0] -60
  INT userid;               ! [1] current user ID of TACL;
                            !     0 if logged off or logged on
                            !     as NULL,NULL (0,0)
  INT cipri;                ! [2] current priority of TACL
  INT ciinfile [0:11];      ! [3] IN file of TACL
  INT cioutfile [0:11];     ![15] OUT file of TACL
  INT config^request^type;  ![27] configuration request type
                            !     0 send default configuration
                            !     1 send user configuration
  END;
```

The length of the message is 56 bytes.

Response. The $CMON reply contains configuration information or a display message.

The format of the configuration information message is as follows:

```
STRUCT config^reply;
  BEGIN
  INT replycode;              !  [0] 0
  INT count;                  !  [1] number of INTs that
                              !      follow (currently12);
  INT autologoffdelay;        !  [2] see #GETCONFIGURATION
  INT logoffscreenclear;      !  [3] description in the
  INT remotesuperid;          !  [4] TACL Reference
  INT blindlogon;             !  [5] Manual for more
  INT namelogon;              !  [6] information on these
  INT cmontimeout;            !  [7] parameters.
  INT cmonrequired;           !  [8]
  INT remotecmontimeout;      !  [9]
  INT remotecmonrequired;     ! [10]
  INT nochangeuser;           ! [11]
  INT stoponfemodemerr;       ! [12]
  INT requestcmonuserconfig;  ! [13]
END;
```

The length of the message is 4 bytes plus 2 times the value in the variable *count*. The message length is currently 28 bytes.

The format of the display message is as follows:

```
STRUCT config^text^reply;
  BEGIN
  INT replycode;         !  [0] <> 0
                         !
  STRING                 !
      replytext [0:n];   !  [1] optional message to be
                         !      displayed; maximum of
                         !      132 bytes
  END;
```

The length of the message is 2 bytes plus the *replytext* length in bytes. The length of *replytext* is implied in the reply count used when making a reply. If *count* = 2, no text is displayed.

20 System Messages

A system message is an interprocess message that is sent from the operating system to an application process. System messages are received by an application process through its $RECEIVE file.

There are two procedures you can use to open $RECEIVE: FILE_OPEN_ and OPEN. The OPEN procedure, which is superseded by the FILE_OPEN_ procedure, is supported for compatibility with previous software and should not be used for new development.

When $RECEIVE is opened with FILE_OPEN_, you can choose to receive D-series-format system messages (the default action) or C-series-format system messages. When $RECEIVE is opened with OPEN, you receive C-series-format messages. Some D-series messages supersede one or more C-series messages, while other D-series messages support newer features. If a process requests C-series messages and uses a feature for which there is no C-series message, then the process receives a D-series message. Table 20-1 lists D-series-format system messages in numerical order; Table 20-1 lists C-series-format system messages in numerical order and, for each message, provides the equivalent D-series-format system message. D-series-format system messages that do not have an equivalent C-series-format system message are listed at the end of this table.

Application Conversion

D-series-format system messages are available for converted applications to read from $RECEIVE when using the D-series and later system procedures. See the *Guardian Application Conversion Guide* and the *Guardian Programmer's Guide* for more detailed information.

Error Return Conventions

The completion of a read associated with a C-series-format system message returns a condition code of "greater than" (CCG) and file-system error 6 from FILE_GETINFO_.

D-series-format system messages do not use condition codes to indicate an error. Instead, each procedure returns an integer `error` value. If an error condition contains more information than the procedure can return in an integer parameter, the procedure returns additional information in an integer `error-detail` parameter.

For more information about system messages, refer to the *Guardian Programmer's Guide*.

Note. When you read system messages by calling the READUPDATE procedure, you must reply in a corresponding call to REPLY, even if you have no message to return. If your application process is performing message queuing, call FILE_GETRECEIVEINFO, LASTRECEIVE, or RECEIVEINFO immediately following completion of the READUPDATE and pass the message tag back to the REPLY procedure.

The message lengths discussed in this section are subject to change. Do not test the received length for equality to an expected length.

The following table lists the D-series-format system messages in numerical order.

Table 20-1. D-Series-Format System Messages (page 1 of 2)

-2	Processor Down
-3	Processor Up
-10	SETTIME
-11	Power On
-12	NEWPROCESSNOWAIT Completion
-13	System Message Buffer Overrun
-21	3270 Device Status Received
-22	Elapsed Time Timeout
-23	Memory Lock Completion
-24	Memory Lock Failure
-26	Process Time Timeout
-32	Process CONTROL
-33	Process SETMODE
-34	Process RESETSYNC
-35	Process CONTROLBUF
-37	Process SETPARAM
-38	Queued Message Cancellation
-41	Nowait DEVICEINFO2 Completion
-100	Remote Processor Down
-101	Process Deletion: ABEND, STOP, or Processor Down
-102	Nowait PROCESS_LAUNCH_ or PROCESS_CREATE_ Completion
-103	Process Open
-104	Process Close
-105	Break on Device
-106	Device Type Inquiry
-107	Subordinate Name Inquiry
-108	Nowait FILE_GETINFOBYNAME_ Completion
-109	Nowait FILENAME_FINDNEXT_ Completion
-110	Loss of Communication With Network Node
-111	Establishment of Communication With Network Node
-112	Job Process Creation
-113	Remote Processor Up

Table 20-1. D-Series-Format System Messages (page 2 of 2)

-121	Pathsend Dialog Abort
-141	Nowait PROCESS_SPAWN_ Completion
-147	Device Information Inquiry

The following table lists the C-series-format system messages in numerical order and, for each message, provides the equivalent D-series-format system message. D-series-format system messages that do not have an equivalent C-series-format system message are listed at the end of the table.

Table 20-2. C-Series and D-Series System Messages Compared (page 1 of 2)

C-Series System Message		**D-Series System Message**	
-2	Processor Down	-2	Processor Down
-2	Processor Down: Named Process Deletion	-101	Process Deletion: Processor Down
-3	Processor Up	-3	Processor Up
-5	Process Deletion: STOP	-101	Process Deletion: STOP
-6	Process Deletion: ABEND	-101	Process Deletion: ABEND
-8	Change in Status of Network Node	-100	Remote Processor Down
		-110	Loss of Communication With Network Node
		-111	Establishment of Communication With Network Node
		-113	Remote Processor Up
-9	Job Process Creation	-112	Job Process Creation
-10	SETTIME	-10	SETTIME
-11	Power On	-11	Power On
-12	NEWPROCESSNOWAIT Completion	-12	NEWPROCESSNOWAIT Completion
-13	System Message Buffer Overrun	-13	System Message Buffer Overrun
-20	Break on Device	-105	Break on Device
-21	3270 Device Status Received	-21	3270 Device Status Received
-22	Elapsed Time Timeout	-22	Elapsed Time Timeout
-23	Memory Lock Completion	-23	Memory Lock Completion
-24	Memory Lock Failure	-24	Memory Lock Failure
-26	Process Time Timeout	-26	Process Time Timeout
-30	Process Open	-103	Process Open
-31	Process Close	-104	Process Close
-32	Process CONTROL	-32	Process CONTROL

Table 20-2. C-Series and D-Series System Messages Compared (page 2 of 2)

C-Series System Message		**D-Series System Message**	
-33	Process SETMODE	-33	Process SETMODE
-34	Process RESETSYNC	-34	Process RESETSYNC
-35	Process CONTROLBUF	-35	Process CONTROLBUF
-37	Process SETPARAM	-37	Process SETPARAM
-38	Queued Message Cancellation	-38	Queued Message Cancellation
-40	Device Type Inquiry	-106	Device Type Inquiry
-41	Nowait DEVICEINFO2 Completion	-41	Nowait DEVICEINFO2 Completion
	none	-107	Subordinate Name Inquiry
	none	-108	Nowait FILE_GETINFOBYNAME_ Completion
	none	-109	Nowait FILENAME_FINDNEXT_ Completion
	none	-141	Nowait PROCESS_SPAWN_ Completion
	none	-147	Device Information Inquiry

Message Descriptions

This subsection lists the system messages and provides a description of each message.

```
-2  PROCESSOR DOWN
```

Cause. The operating system did not receive an “I’m alive” message from the specified processor which was being monitored with the MONITORCPUS procedure. Compare the named process deletion form of message -2, which is listed separately.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -2
sysmsg[1]          = Processor number
```

Response. The response, if any, is application dependent.

```
-2  PROCESSOR DOWN: NAMED PROCESS DELETION
```

Cause. (C-series-format system message only) The operating system did not receive an “I’m alive” message from the specified processor which was being monitored with the MONITORCPUS procedure. This form of the processor down message is sent to the ancestor of a named process (pair) to indicate that the name has been deleted, that is, the only process running under that name was in the processor that failed.

Format. Although this message has the same number as the preceding message (-2), it is distinguished by the presence of a dollar sign ($) in the second word. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]         = -2
sysmsg[1]         = $process-name
sysmsg[1]         = -1
```

Response. The response, if any, is application dependent.

```
-3  PROCESSOR UP
```

Cause. A processor being monitored with the MONITORCPUS procedure was reloaded.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]        = -3
sysmsg[1]        = Processor number
```

Response. The response, if any, is application dependent.

```
-5  PROCESS DELETION (STOP)
```

Cause. (C-series-format system message only) A call to the process-control STOP procedure deleted a process. The stop message is sent to the ancestor of the process and the ancestor of the job (GMOM).

Format. The two forms of the stop message are:

- If the deleted process was not named, or if one member of a process pair deletes the other member, the operating system sends the following form of the message:

```
sysmsg[0]          = -5
sysmsg[1] FOR 4    = Process ID of deleted process
sysmsg[5]          = Header size (header ends at sysmsg[19])
sysmsg[6] FOR 4    = Process processor time in microseconds
                     (a FIXED value)
sysmsg[10]         = The job ID; 0 if process has no GMOM
sysmsg[11]         = The completion code
sysmsg[12]         = Termination information; 0 if the user
                     did not supply information
sysmsg[13] FOR 4 = The subsystem organization name;
                     for HP products,this is HP
sysmsg[17]         = The subsystem number
sysmsg[18]         = The subsystem version
sysmsg[19]         = The length of text in bytes
sysmsg[20] FOR n = Up to 80 bytes of text
```

- If the call to the STOP procedure deletes the process name from the process-pair directory, the operating system sends this message:

  ```
  sysmsg[0]          = -5
  sysmsg[1]  FOR 3   = The name of the deleted process
  sysmsg[4]          = -1
  sysmsg[5]          = The header size (header ends at
                       sysmsg[19])
  sysmsg[6]  FOR 4   = Process processor time in microseconds
                       (a FIXED value)
  sysmsg[10]         = The job ID; 0 if process has no GMOM
  sysmsg[11]         = The completion code
  sysmsg[12]         = Termination information, 0 if the user
                       did not supply information
  sysmsg[13]  FOR 4 = Subsystem organization (8 bytes);
                       for HP products, this is HP
  sysmsg[17]         = Subsystem number
  sysmsg[18]         = Subsystem version
  sysmsg[19]         = Length of text in bytes
  sysmsg[20]  FOR n = Text (up to 80 bytes)
  ```

  If an external process caused the termination, the stop message is changed as follows:

  ```
  sysmsg[11]         = Completion code defaults to 6
  sysmsg[12]         = Creator access ID
  sysmsg[13]  FOR 4 = Process ID of the process that caused
                       the termination
  ```

  This message indicates that neither member of the process pair exists.

For information about completion codes, refer to Table 20-3 on page 20-21.

Response. The response, if any, is application dependent.

```
-6  PROCESS DELETION (ABEND)
```

Cause. (C-series-format system message only) A process was deleted because of a call to the process-control ABEND procedure, or because the deleted process encountered a trap condition and was aborted by the operating system.

The abend message is sent to the ancestor of the process and the ancestor of the job (GMOM).

Format. The two forms of the abend message are:

- If one member of the process pair is deleted or if the deleted process was not named, the operating system sends the following message:

```
sysmsg[0]          = -6
sysmsg[1]  FOR 4   = Process ID of deleted process
sysmsg[5]          = Header size (header ends at
                     sysmsg[19])
sysmsg[6]  FOR 4   = Process processor time in microseconds
                     (a FIXED value)
sysmsg[10]         = The job ID; 0 if process has no GMOM
sysmsg[11]         = The completion code
sysmsg[12]         = Termination information; 0 if the user
                     did not supply information
sysmsg[13]  FOR 4  = The subsystem organization name;
                     for HP products, this is HP
sysmsg[17]         = The subsystem number
sysmsg[18]         = The subsystem version
sysmsg[19]         = The length of text in bytes
sysmsg[20]  FOR n  = Up to 80 bytes of text
```

- If the operating system deletes the process name from the process-pair directory, it sends the following message. This message indicates that neither member of the process pair exists.

```
sysmsg[0]         = -6
sysmsg[1]  FOR 3  = The name of the deleted process
sysmsg[4]         = -1
sysmsg[5]         = The header size (header ends at
                    sysmsg[19])
sysmsg[6]  FOR 4  = Process processor time in microseconds
                    (a FIXED value)
sysmsg[10]        = The job ID; 0 if process has no GMOM
sysmsg[11]        = The completion code
sysmsg[12]        = Termination information; 0 if the user
                    did not supply information
sysmsg[13] FOR 4 = Subsystem organization (8 bytes);.
                    for HP products,this is HP
sysmsg[17]        = Subsystem number
sysmsg[18]        = Subsystem version
sysmsg[19]        = Length of text in bytes
sysmsg[20] FOR n = Text (up to 80 bytes)
```

- If an external process caused the termination, the abend message is changed as follows:

```
sysmsg[11]       = Completion code defaults to 6
sysmsg[12]       = Creator access ID
sysmsg[13] FOR 4 = Process ID of the process that caused
                   the termination
```

- If a trap is encountered, the abend message is changed as follows:

```
sysmsg[11]        = Completion code is -1
sysmsg[12]        = Termination information is 0
sysmsg[13] FOR 4  = Subsystem organization is 4 words of
                    blanks
sysmsg[17]        = Subsystem number is 0
sysmsg[18]        = Subsystem version is 0
sysmsg[19]        = Length is 76 bytes of text
sysmsg[20] FOR 76 = Is the following text:

TRAP NO=nn, S=nnnnnn, CS=nn, P=nnnnnn,
ENV=nnnnnn, L=nnnnnn, OCT P=nnnnnnnn
```

For information about the completion codes, refer to Table 20-3 on page 20-21.

Response. The response, if any, is application dependent.

```
-8  CHANGE IN STATUS OF NETWORK NODE
```

Cause. (C-series-format system message only) The process that was running on a system that is part of a network enabled receipt of remote status-change messages by passing “1” as a parameter to the MONITORNET procedure.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]         = -8
sysmsg[1].<0:7>   = System number
sysmsg[1].<8:15>  = Number of processors
sysmsg[2]         = Current processor-status bit mask
sysmsg[3]         = Previous processor-status bit mask
```

Response. The response, if any, is application dependent.

```
-9  JOB PROCESS CREATION
```

Cause. (C-series-format system message only) The receiving process is the supervisor of a job and a process running under the supervisor's GMOM job ID created a third process.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]        = -9
sysmsg[1] FOR 4 = Job ID
sysmsg[5]        = Process ID of the newly created process
```

Response. The response, if any, is application dependent.

```
-10  SETTIME
```

Cause. The system manager or operator reset the processor's internal clock.

If a call to MONITORNEW enabled receipt of SETTIME messages, the operating system sends the process the following message.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]        = -10
sysmsg[1]        = Processor number
sysmsg[2] FOR 4 = Signed change in microseconds
                   (FIXED integer)
sysmsg[6]        = Reason code
```

The reason codes are:

Code	Meaning
0	Initial setting (Greenwich mean time (GMT) and local civil time (LCT) change).
1	Subsequent adjustment (GMT and LCT time change).
2	Daylight-savings time (LCT time change) transition.

Response. The response, if any, is application dependent.

```
-11  POWER ON
```

Cause. The processor power failed, then returned.

If a call to MONITORNEW enabled receipt of power on messages, the operating system sends the process the following message.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]            = -11
sysmsg[1]            = Processor number
```

Response. The response, if any, is application dependent.

```
-12  NEWPROCESSNOWAIT COMPLETION
```

Cause. A call to the NEWPROCESSNOWAIT procedure finished.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -12
sysmsg[1].<0:7>    = NEWPROCESS error
sysmsg[1].<8:15>   = NEWPROCESS suberror or file-system
                     error
sysmsg[2] FOR 2    = Tag passed to NEWPROCESSNOWAIT in
                     filenames [36:37]
sysmsg[4] FOR 4    = Process ID of the new process
sysmsg[8]          = NEWPROCESS error
sysmsg[9]          = NEWPROCESS suberror or file-system
                     error
```

For more information on the NEWPROCESS error, refer to Section 5, NEWPROCESS AND NEWPROCESSNOWAIT Errors.

Note. If 119 is returned in *sysmsg*[1].<8:15>, the file-system error number is greater than 255. The actual error is in *sysmsg*[9].

Response. The response, if any, is application dependent.

```
-13  SYSTEM MESSAGE BUFFER OVERRUN
```

Cause. Some broadcast system messages were not delivered to this process's $RECEIVE file because the process did not read $RECEIVE as fast as the system buffer was filled. Broadcast messages are messages that can go to every process (system messages -2, -3, -8, -10, and -11).

The system delivers all broadcast messages generated from this time forward.

Format. *sysmsg* [0] = -13

Response. The response, if any, is application dependent.

```
-20  BREAK ON DEVICE
```

Cause. (C-series-format system message only) The BREAK key was pressed on a monitored terminal, or a process called SENDBREAKMESSAGE.

If the process specified break monitoring through a call to SETMODE or SETMODENOWAIT, the operating system sends the process the following message.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]  = -20
sysmsg[1]  = Logical device number, in binary, of device
             where BREAK was pressed.  If a process called
             SENDBREAKMESSAGE, this field contains -1.
sysmsg[2]  = System number, in binary, of logical device
             number or SENDBREAKMESSAGE caller.
sysmsg[3]  = The most significant word of the break tag.
sysmsg[4]  = The least significant word of the break tag.
```

Response. The response, if any, is application dependent.

```
-21  3270 DEVICE STATUS RECEIVED
```

Cause. A call to SETMODE 53 was made by the application to monitor subdevice status, pass the information to TR3271 by way of a call to SETMODE 51, and issue this status message.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0] = -21
sysmsg[1] = The response ID
sysmsg[2] = The actual 3271 status bytes:
            .<0:7> = sense byte
            .<8:15> = status byte
sysmsg[3] = A translation of the device status to status
            bits; the application might pass
            this word directly to TR3271 by way of SETMODE
            51 to post the status on a TR3271 subdevice.
```

Response. The response, if any, is application dependent.

```
-22  ELAPSED TIME TIMEOUT
```

Cause. A timer set by a call to SIGNALTIMEOUT timed out.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]       = -22
sysmsg[1]       = parameter1 supplied to SIGNALTIMEOUT
                  (0 if none)
```

```
sysmsg[2] FOR 2 = parameter2 supplied to SIGNALTIMEOUT
                  (0D if none)
```

Response. The response, if any, is application dependent.

```
  -23  MEMORY LOCK COMPLETION
```

Cause. A call to the privileged procedure LOCKMEMORY waited for memory but completed successfully before the specified time limit was reached.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -23
sysmsg[1]          = parameter1 supplied to LOCKMEMORY
sysmsg[2] FOR 2    = parameter2 supplied to LOCKMEMORY
                     (if none supplied, 0D)
```

Response. The response, if any, is application dependent.

```
  -24  MEMORY LOCK FAILURE
```

Cause. A call to the privileged procedure LOCKMEMORY waited for memory but timed out without completing the lock.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -24
sysmsg[1]          = parameter1 supplied to LOCKMEMORY
                     (if none supplied, 0)
sysmsg[2] FOR 2    = parameter2 supplied to LOCKMEMORY
                     (if none supplied, 0D)
```

Response. The response, if any, is application dependent.

```
  -26  PROCESS TIME TIMEOUT
```

Cause. A timer set by a call to SIGNALPROCESSTIMEOUT timed out.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -26
sysmsg[1]          = parameter1 supplied to
                     SIGNALPROCESSTIMEOUT (0 if none)
sysmsg[2] FOR 2    = parameter2 supplied to
                     SIGNALPROCESSTIMEOUT (0D if none)
```

Response. The response, if any, is application dependent.

```
-30  PROCESS OPEN
```

Cause. (C-series-format system message only) Either the process was opened by another process or the backup process of a process pair opened a process (a process receives two process open messages when opened by a process pair).

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]         = -30
sysmsg[1]         = flags parameter to caller's OPEN
sysmsg[2]         = sync-or-receive-depth  parameter to
                    caller's OPEN
sysmsg[3] FOR 4   = 0 if normal OPEN; process ID of primary
                    process if an open by a backup process
sysmsg[7]         = 0 if normal OPEN, negative of the file
                    number of file if an open by a backup
                    process
sysmsg[8]         = Process accessor ID of opener
sysmsg[9] FOR 4   = Optional first qualified name of named
                    process or blanks
sysmsg[13] FOR 4  = Optional second qualified name of named
                    process or blanks
sysmsg[17].<14>   = Set to 1 if the opener's process access
                    identification (given in the message)
                    has not been verified in the receiver's
                    node (although it passes the
                    remote password test)
sysmsg[17].<15>   = Set to 1 if the opener is on a
                    different node from the receiving
                    process
```

Response. Obtain the process ID of the opener by a call to LASTRECEIVE or RECEIVEINFO. Corrective action, if any, is application dependent.

```
-31  PROCESS CLOSE
```

Cause. (C-series-format system message only) Another process closed the receiver process. The closing process can also be the backup process of a process pair, so a process receives two process close messages when closed by a process pair.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]  = -31
```

Response. Obtain the process ID of the closer by a call to LASTRECEIVE or RECEIVEINFO. Corrective action, if any, is application dependent.

```
-32  PROCESS CONTROL
```

Cause. Another process called the CONTROL procedure while referred to the receiver process file.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]      = -32
sysmsg[1]      = Operation parameter to caller's CONTROL
sysmsg[2]      = Parameter parameter to caller's CONTROL
```

Response. Obtain the process ID of the caller to CONTROL by a call to LASTRECEIVE or RECEIVEINFO. The response, if any, is application dependent.

```
-33  PROCESS SETMODE
```

Cause. Another process referred to the receiving process in a call to the SETMODE or SETMODENOWAIT procedure.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]     = -33
sysmsg[1]     = function parameter to caller's SETMODE or
                SETMODENOWAIT
sysmsg[2]     = param1 parameter to caller's SETMODE
                or SETMODENOWAIT
sysmsg[3]     = param2 parameter to caller's SETMODE
                or SETMODENOWAIT
```

If the receiving process can handle requests for last parameter information (by setting *param1*.<15> in a SETMODE 80 call), the flags word is included:

```
sysmsg[4] = Flags word:
           .<13> = 1 if param1 was supplied in the call to
                   SETMODE
           .<14> = 1 if param2 was supplied in the call
                   to SETMODE
           .<15> = 1 if last-params was supplied in the
                   call to SETMODE
```

Response. Obtain the process ID of the caller to SETMODE or SETMODENOWAIT by a call to LASTRECEIVE or RECEIVEINFO. The response, if any, is application dependent.

A process that receives the extended (five-word) SETMODE message can return a value to be placed in the *last-params* parameter of the SETMODE caller. The process should call REPLY with a buffer that has the format:

```
replymsg[0]   = -33
replymsg[1]   = The previous value of param1
replymsg[2]   = The previous value of param2
```

Supplying a reply when none is needed will not cause an error.

```
-34  PROCESS RESETSYNC
```

Cause. Another process called the RESETSYNC procedure while referring to the receiver process file. A call to the CHECKPOINT procedure might contain an implicit call to RESETSYNC.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]  = -34
```

Response. Obtain the process ID of the caller to RESETSYNC by a call to LASTRECEIVE or RECEIVEINFO. A server process using the sync ID mechanism should clear its local copy of the sync ID value. Corrective action, if any, is application dependent.

```
-35  PROCESS CONTROLBUF
```

Cause. Another process called the CONTROLBUF procedure while referring to the receiver process file.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]         = -35
sysmsg[1]         = operation parameter to the caller's
                    CONTROLBUF
sysmsg[2]         = count parameter to the caller's
                    CONTROLBUF
sysmsg[3] FOR n   = buffer data from caller's CONTROLBUF,
                    where n is the number of words in
                    buffer
```

Response. Obtain the process ID of the caller to CONTROLBUF by a subsequent call to LASTRECEIVE or RECEIVEINFO. The response, if any, is application dependent.

```
 -37  PROCESS SETPARAM
```

Cause. Another process referred to the receiving process in a SETPARAM call.

The operating system sends this message only if the receiving process has indicated it will accept such messages by doing one of the following:

- Omitting the FILE_OPEN_ procedure *options* parameter, or setting *options*.<15> = 0
- Setting *flags*.<1> = 1 in its call to OPEN on $RECEIVE
- Setting *param1*.<14> to 1 in a call to SETMODE 80

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]     = -37
sysmsg[1]     = function parameter to SETPARAM
sysmsg[2]     = Flags word:
                .<14> = 1 if param-array was supplied in
                        the call to SETPARAM
                .<15> = 1 if last-param-array was supplied
                        in the call to SETPARAM
sysmsg[3]     = param-count parameter specified in the call
                to SETPARAM
sysmsg[4:n]   = param-array parameter to SETPARAM with a
                length of param-count bytes
```

Response. To return a value in the *last-param-array* parameter of the SETPARAM caller, the receiving process should call REPLY with a buffer that has the following format:

```
replymsg[0]   = -37
replymsg[1]   = The value in the range 0 through 256 for
                last-param-count
replymsg[2:n] = The value for last-param-array, with a
                length of last-param-count bytes
```

If the format of the reply is incorrect, error code 2 is returned to both the caller of REPLY and the caller of SETPARAM.

```
  -38  QUEUED MESSAGE CANCELLATION
```

Cause. A pending message was canceled. (A pending message is a message read by READUPDATE but not yet replied to.)

This message can occur at random relative to other messages. For example, if an opener calls CLOSE or FILE_CLOSE_ while it has an outstanding operation, the operational message is canceled and the close message is sent. The receiving process cannot assume that the cancellation message appears on $RECEIVE before the close message (or vice versa).

The absence of a cancellation message does not mean that the sending process received a reply. A cancellation can occur at any time, including just before a reply or while the reply data is awaiting delivery; in these cases, the cancellation message does not appear.

Do not use the cancellation notice to determine the delivery status of replies.

Format. Message -38 can be received only if SETMODE 80 *param*.<13> has been set to 1:

```
sysmsg[0]   = -38
sysmsg[1]   = The message-tag of the canceled message
```

Response. Call REPLY for the referenced message.

```
  -40  DEVICE TYPE INQUIRY
```

Cause. (C-series-format system message only) Another process called DEVICEINFO, DEVICEINFO2, or FILEINFO to request the device type or the physical record length from the receiving process. This message is sent only to subtype 30 processes. Do not attempt to use subtype 30 processes to emulate disk devices (device type 3). Calls related to disk files (such as FILEINFO) return file-system error 2 even if the process is emulating a disk file.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]      = -40
sysmsg[1:4]    = The first qualifier name in internal
                 format (blank padded)
sysmsg[5:8]    = The second qualifier name in internal
                 format (blank padded)
```

If the calling process did not specify qualifiers to the process name, *sysmsg*[1:4] and *sysmsg*[5:8] are blank.

Response. The subtype 30 process should call REPLY with the needed information in a buffer of the following format:

```
replymsg[0]   = -40
replymsg[1]   = The device type word:
                .<0:3>   = 0
```

```
               .<4:9>   = device type
               .<10:15> = device subtype
replymsg[2]  = Physical record length
```

If the message response is incorrectly formatted, the DEVICEINFO[2] or FILEINFO caller receives the default values (device type and subtype of zero) and the REPLY caller receives error code 2.

```
-41  NOWAIT DEVICEINFO2 COMPLETION
```

Cause. The DEVICEINFO2 nowait option (*options*.<13>) was specified to obtain device information. The device information is returned only in system message -41 on $RECEIVE.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]   = -41
sysmsg[1:2] = Tag value from tag-or-timeout parameter
sysmsg[3]   = File-system error code
sysmsg[4]   = The devtype value
sysmsg[5]   = The physical-recordlen value
sysmsg[6]   = The diskprocess-version value
```

The device information returned in words 4 through 6 is valid only if the file-system error code (word 3) is zero. Words 1 and 2 return the value passed to DEVICEINFO2 to help you identify this particular completion when there are simultaneous inquiries.

Response. The response, if any, is application dependent.

```
-100  REMOTE PROCESSOR DOWN
```

Cause. The remote operating system has declared a processor to be down, and the process had called MONITORNET to monitor remote status changes.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -100
sysmsg[1] FOR 2    = Node number
sysmsg[3]          = Processor number
sysmsg[4]          = Length of node name, in bytes
sysmsg[5] FOR 3    = Reserved
sysmsg[8] FOR *    = Node name (including the \)
```

This message is not sent if the last processor in a node fails; that case is reported as a node failure (message -110).

Response. The response, if any, is application dependent.

```
-101  PROCESS DELETION
```

Cause. A process (pair) has terminated; the cause of termination is indicated in the message. This message is sent:

- To the mom process if it exists and termination was not caused by processor failure
- To the GMOM (NetBatch job ancestor) if it exists and termination was not caused by processor failure
- To the ancestor if the terminated process was a named process without a backup. In this case, the message is sent even if the process is terminated by a processor failure.

If the same process fulfills multiple roles (for example, ancestor and GMOM), then it receives only one copy of the deletion message.

If the terminated process was created by the NEWPROCESS procedure, or if the "send to any ancestor" create option (`create-options`.<9> in the PROCESS_CREATE_ call) was used, the deletion message is delivered to any process with the same name as the original ancestor, regardless of sequence number. Otherwise, the deletion message is delivered only to the original instance of a named ancestor process, as indicated by having the original sequence number. All the processes in an unbroken string of primary and backup processes with the same name are considered part of the same instance and have the same sequence number.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -101
sysmsg[1] FOR 10   = Process handle of terminated process
sysmsg[11] FOR 4   = Process processor time in microseconds
                     (a FIXED value)
sysmsg[15]         = Process job ID; 0 if the process is
                     not part of a job
sysmsg[16]         = Completion code
sysmsg[17]         = Termination information (0 if none
                     supplied)
```

The next three items contain the SPI subsystem ID:

```
sysmsg[18] FOR 4   = Subsystem organization: "HP" for
                     HP supplied subsystems; blank if
                     not supplied
sysmsg[22]         = Subsystem number: zero if not supplied
sysmsg[23]         = Subsystem version: zero if not supplied
sysmsg[24] FOR 10  = Process handle of external process
                     causing termination; null process
                     handle (all words set to -1) if none
sysmsg[34]         = Length in bytes of termination text
sysmsg[35]         = Offset in bytes (from beginning of
                     message) of process descriptor
                     of terminated named process (pair)
sysmsg[36]         = Length in bytes of process descriptor
                     of terminated named process (pair)
sysmsg[37].<0:13>  = Reserved
sysmsg[37].<14>    = OSS system type: 1 if the terminated
                     process was an OSS process; 0 if
                     the terminated process was a Guardian
                     process
sysmsg[37].<15>    = Abend: termination caused by ABEND if 1,
                     STOP if 0
sysmsg[38] FOR 2   = OSS process ID
sysmsg[40]         = Reserved
sysmsg[41] FOR *   = Termination text (80-byte limit). The
                     length of this item is in sysmsg[34].
                     It is zero length if no text was
                     supplied.
sysmsg[  ] FOR *   = Process descriptor of terminated named
                     process (pair).  The offset of this
                     item is in sysmsg[35].  The length of
                     this item is in sysmsg[36].  If the
                     terminated process is unnamed or is
                     the recipient's backup, this is zero
                     length.
```

The returned process descriptor for a terminated named process (pair) is in the form:

`\`*`node`*`.$`*`name`*`:`*`seqno`*

Table 20-3 shows HP completion codes; we recommend that you use positive completion codes in the same way. You can specify a completion code with any positive value in a PROCESS_STOP_, STOP, or ABEND procedure call. Negative completion codes are reserved for HP use.

For an OSS process terminating as a result of an `exit()` function call, the completion code is set to the low-order 8 bits of the exit status.

Table 20-3. Completion Codes (page 1 of 4)

Completion Code	Definition
0	Normal, voluntary termination with no errors. This code is the default for PROCESS_STOP_ (if abnormal termination is not specified) and STOP if no completion code is specified, and for the OSS `exit()` function if no exit status is specified.
1	Normal, voluntary termination with WARNING diagnostics. For example, if the process is a compiler, the compilation terminated with WARNING diagnostics after building a complete object file.
2	Abnormal, voluntary termination with FATAL errors or diagnostics. For example, if the process is a compiler, the compilation terminated with FATAL diagnostics and either an object file was not built or, if built, might be incomplete. A complete listing is generated.
3	Abnormal, voluntary, but premature termination with FATAL errors or diagnostics. For example, if the process is a compiler, the compilation terminated with FATAL diagnostics, with either no object file or an incomplete object file being built and an incomplete listing generated (the compiler quit compiling prematurely).
4	Process never got started. This completion code exists primarily for the use of the command interpreter or other command language interpreters that can act as the executor process of a batch job. This code allows the executor process to detect that a process associated with a RUN statement never got started. In that sense, this completion code is a “fake” completion code. The command interpreter acts as though it received a termination message from the process that it tried to create, when in fact it received an error returned by the procedure or OSS function that launched the process. The command interpreter then makes the completion code and the error returned by the procedure or OSS function that launched the process available for evaluation, for example, by a batch job executor process.
5	Process calls PROCESS_STOP_ (with abnormal termination specified) or ABEND on itself. This code is the default completion code for the PROCESS_STOP_ procedure (when abnormal termination is specified) and the ABEND procedure.

Table 20-3. Completion Codes (page 2 of 4)

Completion Code	Definition
6	PROCESS_STOP_, STOP, or ABEND was called to delete a process by an external, but authorized, process. The system includes this completion code in the process deletion message. If the process cannot be stopped, the request is saved so that when the process calls SETSTOP this completion code is sent with the process deletion message. The user ID, the PCBCRAID (CAID) and the process ID of the process that caused the termination, are included in the termination message.
7	Restart this job. This completion code is used by the NetBatch scheduler and an executor process. The executor process sets its completion code to this value upon termination; the scheduler interprets this completion code and restarts a "restartable" job.
8	Code 8 is the same as code 1, normal termination, except that the user must examine the listing file to determine whether the results are acceptable. Completion code 8 is typically used by compilers.
9	The `kill()` or `raise()` OSS function generated a signal that stopped the process. The termination information provides the signal number. Note that if a signal is delivered to a signal handler that stops the process, the completion code will be determined by the handler. For example, when a signal stops a TNS/R native C program, a different completion code is returned as set by the signal handler installed by the Common Run-Time Environment (CRE).
-1	A trap was detected in a Guardian TNS process. If the system detects the absence of a trap handler routine or encounters another trap in a trap handler, then in addition to an abnormal termination, this completion code is returned automatically in the process deletion (ABEND) message. The contents of the text string vary with the state of the process. The first nine characters are "TRAPNO=*nn*" with *nn* representing the trap number in decimal. Then the text identifies the code space, including the TNS code segment index when appropriate, and indicates whether the process was privileged. Finally, the text displays key registers, depending upon the execution mode of the process at the time of its termination: P or pc, L, and S for TNS or accelerated mode; pc and sp for TNS/R native mode. Examples: Invalid address in TNS mode: TRAPNO=00: (UC.00) P=%000012 L=%000001 S=%000003 Instruction failure in accelerated mode: Arithmetic overflow (division by zero) in accelerated mode, privileged: TRAPNO=02: (acc UC, Priv) pc=%h7042370C L=%023520 S=%023526 Limits-exceeded in TNS/R native mode, privileged: TRAPNO=05: (SCr, Priv) pc=0x808E2EDC sp=0x5FFFFF00

Table 20-3. Completion Codes (page 3 of 4)

Completion Code	Definition
-2	This code is returned by the system when a process has terminated itself but the system is unable to pass along the requested completion code and the associated termination information due to a resource problem in the system.
-3	This code is returned by the system when a process terminating itself passed bad parameters to PROCESS_DELETE_, STOP, or ABEND. In this case, some or all of the information requested in the completion code message may not be present. Since the process is stopping itself, it is stopped.
-4	This code is returned by the system when a processor failure caused the name of a process to be deleted (that is, the only process running under that name was in the processor that failed).
-5	A communications or resource failure occurred during the execution of one of the OSS `exec` set of functions; or an initialization failure of the new process occurred when it was too late for the `exec` set of functions to return an error to its caller.
-6	An OSS process or TNS/R native process terminated when it caused a hardware exception. The termination information field of the message contains the signal number. The termination text is in the message for all processes. However, while the TACL command interpreter displays the termination text when it is present in the message for a process created by TACL, OSS utilities such as `osh` typically do not. The text shows the signal number and name, identifies the code space, and indicates whether the process was privileged. For a TNS/R native process, the text displays the pc and sp registers. For an OSS process, it shows registers appropriate to the mode, as for completion code -1. Examples: Invalid address in TNS/R native mode: Signal 11, SIGSEGV: (UCr) pc=0x700024F0 sp=0x4FFFFC68
-7	An OSS process or TNS/R native process terminated as a result of a corrupted stack frame or register state.
-8	An OSS process or TNS/R native process terminated because of insufficient user stack space for signal delivery. Stack overflow generates completion code -8, which is otherwise like completion code -6. Example: Stack overflow in TNS/R native mode: Signal 25, SIGSTK: (UCr) pc=0x70000394 sp=0x4FEFFE18
-9	An OSS process or TNS/R native process terminated because of insufficient PRIV stack space for signal delivery. The termination information field of the message contains the signal number.

Table 20-3. Completion Codes (page 4 of 4)

Completion Code	Definition
-10	An OSS process or TNS/R native process terminated because it was unable to obtain resources for signal delivery. The termination information field of the message contains the signal number.
-11	An OSS process or TNS/R native process terminated because it attempted to resume from a nonresumable signal. The termination information field of the message contains the signal number.
-12	One of the OSS `exec` or `tdm_exec` set of functions executed successfully. The OSS process ID continues to exist as it migrates to another process handle, but the original process handle is deleted. Call PROCESS_GETINFOLIST_ to obtain the new process handle of the OSS process.
-13	The OSS `open()` or `dup()` function performed by the PROCESS_SPAWN_ procedure failed. The termination information in *sysmsg*[17] contains the OSS `errno` for the error that occurred. The subsystem ID in *sysmsg*[18] contains the null value. The termination text in *sysmsg*[41] can contain additional information.

Response. The response, if any, is application dependent.

```
-102  NOWAIT PROCESS_LAUNCH_ OR PROCESS_CREATE_ COMPLETION
```

Cause. A nowait call to the PROCESS_LAUNCH_ OR PROCESS_CREATE_ procedure finished.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]           = -102
sysmsg[1] FOR 2     = nowait-tag supplied to PROCESS_CREATE_
sysmsg[3] FOR 10    = process-handle of new process
sysmsg[13]          = error
sysmsg[14]          = error-detail
sysmsg[15]          = Length in bytes of process descriptor
                      of new process
sysmsg[16] FOR 4    = Reserved
sysmsg[20] FOR *    = Process descriptor of new process
                      (the length of this item is in
                      sysmsg[15])
```

If the new process is unnamed, then the returned process descriptor is in unnamed form:

`\node.$:cpu:pin:seqno`

If the new process is named, then the returned process descriptor is in named form:

`\node.$name:seqno`

When a named process creates its backup, the returned process descriptor is in named form.

If this message returns an error value indicating that process creation failed, then it does not return a valid process handle or a process descriptor; the process descriptor length is 0.

Response. The response, if any, is application dependent.

```
 -103  PROCESS OPEN
```

Cause. The receiving process was opened by another process.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]          = -103
sysmsg[1]          = Access mode (from the access
                     FILE_OPEN_ parameter)
sysmsg[2]          = Exclusion mode (from the exclusion
                     FILE_OPEN_ parameter)
sysmsg[3]          = Nowait depth (from the nowait
                     FILE_OPEN_ parameter)
sysmsg[4]          = Sync depth (from the
                     sync-or-receive-depth FILE_OPEN_
                     parameter)
sysmsg[5]          = Open options (from the options
                     FILE_OPEN_ parameter)
sysmsg[6]          = User ID of opener (process access ID)
sysmsg[7]          = Miscellaneous:
                     .<0:12> = currently undefined; subject
                               to change
                     .<13> = the opener's user ID has not
                               been verified locally (in the
                               receiver's node0); in any case
                               it will have passed a remote
                               password check
                     .<14> = the opener is on a different
                               node from the receiver.
                     .<15> = backup open: this is an open
                               by a backup
sysmsg[8] FOR 10   = For a backup open, this contains the
                     process handle of the primary process;
                     for a normal open, this is a null
                     process handle
sysmsg[18]         = The length in bytes of the qualifier
                     name given below
sysmsg[19]         = The offset in bytes from the beginning
                     of the message to the beginning of the
                     opener process name appearing below
sysmsg[20]         = The length in bytes of the opener
                     process name appearing below
sysmsg[21]         = For a backup open, the file number
                     used by the primary.  It is typically,
                     but not always, the same as the
                     backup. Unlike the old open message,
                     this value is never negative.
sysmsg[22]         = The creator access ID of the opener.
                     Unlike the process access ID given
                     earlier in the message, this ID is not
```

```
                          verified by remote password checking.
sysmsg[23]              = The offset in bytes from the beginning
                          of the message to the beginning of the
                          opener home terminal name
sysmsg[24]              = The length in bytes of the opener home
                          terminal name appearing below
sysmsg[25] FOR 5        = Reserved, subject to change
sysmsg[30] FOR *        = The qualifier portion of the name used
                          to open the process, in external form
                          (for example, "#PORT2.CTL").  The
                          length of this item is in sysmsg[18].
                          It is zero length if no qualifier was
                          given.
sysmsg[  ] FOR *        = For a named opener, this is the
                          process name in external process
                          descriptor form (system, name, and
                          sequence number).  If the opening
                          process is unnamed, the length is
                          zero.  In either case, the  opener's
                          process handle is available from
                          FILE_GETRECEIVEINFO_.  The offset and
                          length of the name are in sysmsg[19]
                          and sysmsg[20].
sysmsg[  ] FOR *        = The opener's home terminal name in
                          external form.  This is zero
                          length if the opening operating system
                          version is earlier than C10.  The
                          offset and length of the name are in
                          sysmsg[23] and sysmsg[24].
```

Response. If the receiver wishes to reject the open, it should call REPLY with an appropriate file-system error code (> 9). If the application is tracking openers, the opener identified from FILE_GETRECEIVEINFO_ should be added to its table and, in the case of a backup open, associated with the primary open. If the application wishes to have a particular value (typically an open table index) returned in the OpenLabel field from FILE_GETRECEIVEINFO_ on later messages from this opener, it should call REPLY with the desired value in the reply data buffer in the following form:

```
repmsg[0]  = -103
repmsg[1]  = OpenLabel value
```

```
-104  PROCESS CLOSE
```

Cause. The receiving process was closed by another process.

Format. The operating system sends the process the following message, provided the process has opened its $RECEIVE file to receive file management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg[0]  = -104
sysmsg[1]  = tapedisposition parameter to FILE_CLOSE_
```

Response. If the application is tracking openers, the receiver should call FILE_GETRECEIVEINFO_ to obtain the sender's process handle, which can then be used to search the application's tables for the entry to be deleted.

```
-105  BREAK ON DEVICE
```

Cause. The BREAK key was pressed on a terminal or other device for which the application has enabled break monitoring.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -105
sysmsg[1]          = File number of the receiver's open
                     file to the terminal that indicated
                     break (or -1 if unavailable).
                     Before D00, this is the file number
                     to that device (or, if there is more
                     than one, it can be the number of
                     any of the files). The device must
                     be open.
sysmsg[2] FOR 2    = The break tag value specified with
                     SETPARAM (if used)
```

Response. The application should take action in one of the ways described in the *Guardian Programmer's Guide*.

```
 -106  DEVICE TYPE INQUIRY
```

Cause. The receiving process (which must have subtype 30 to receive this message) was the subject of a call to FILE_GETINFOBYNAME_, DEVICEINFO, or a similar function requesting device-type information.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]           = -106
sysmsg[1] FOR 3     = Reserved
sysmsg[4]           = Length in bytes of the qualifier part
                      of the file name being inquired about
                      (zero if none given)
sysmsg[5] FOR *     = The qualifier part of the file name
                      being inquired about, in external
                      form (the length is given in the
                      previous field)
```

Response. The subtype 30 process should REPLY with the needed information in the following form:

```
repmsg[0]           = -106
repmsg[1]           = Device type
repmsg[2]           = Device subtype
repmsg[3] FOR 3     = Reserved, must be filled with -1
repmsg[6]           = Physical record length
```

```
-107  SUBORDINATE NAME INQUIRY
```

Cause. The receiving process (which must have requested these messages by an explicit call to PROCESS_SETINFO_) is being queried for subordinate names by another process calling FILENAME_FINDNEXT_.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]           = -107
sysmsg[1]           = Flags:
                      .<0:14> = Currently undefined; subject
                                to change
                      .<15> = Skip if same; if set and the
                                start name given below
                                exists, skip it and return
                                the following name.  If
                                equal to 0, return the start
                                name if it exists.
sysmsg[2]           = Length in bytes of the start name
                      (zero if no start name supplied)
sysmsg[3]           = The offset in bytes from the beginning
                      of the message to the beginning of the
                      pattern appearing below
sysmsg[4]           = Length in bytes of the pattern
sysmsg[5] FOR 3     = Reserved, subject to change
sysmsg[8] FOR *     = The start name: the qualifier part of
                      the file name, in external form, at
                      which to start searching for a name to
                      sysmsg[2]
sysmsg[  ] FOR *    = The pattern: the qualifier part of
                      the pattern for which a name is to be
                      returned (the offset and length are
                      given above)
```

Response. The process should search its list of subordinate names, starting with (or after, if the Skip If Same option is used) the start name (or from the beginning, if the start name is zero length). The process should return the first name that matches the pattern (as determined by FILENAME_MATCH_). If there are no matching names, a REPLY with error 1 should be made. When replying with a name, subtype 30 processes must supply more information than normal processes, as indicated below; for normal processes, these fields are ignored. To return a matching name, the process should REPLY with a data buffer in the following form:

```
repmsg[0]           = -107
repmsg[1]           = For subtype 30 processes: device type
repmsg[2]           = For subtype 30 processes: device
                      subtype
repmsg[3] FOR 3     = For subtype 30 processes: reserved,
                      must be filled with -1
```

```
repmsg[6]          = Returned name length in bytes
repmsg[7] FOR *    = Returned name (external form
                     qualifier, as for example
                     "#PORT1.CTL")
```

```
-108  NOWAIT FILE_GETINFOBYNAME_ COMPLETION
```

Cause. The receiving process called FILE_GETINFOBYNAME_ specifying that the information be obtained in a nowait manner, and the information is now available.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -108
sysmsg[1] FOR 2    = The tag value from the timeout-or-tag
                     parameter to FILE_GETINFOBYNAME_
sysmsg[3]          = The file-system error code giving the
                     resultant status of the request
sysmsg[4] FOR 5    = The type information (device type,
                     and so on) having the same layout as
                     described for the typeinfo parameter
                     of FILE_GETINFOBYNAME_
sysmsg[9]          = The physical record length value
sysmsg[10]         = The flag value from the flags
                     parameter to FILE_GETINFOBYNAME_
                     .<0:14> = Currently undefined; subject
                               to change
                     .<15> = 0 signifies a Guardian file
                     .<15> = 1 signifies an OSS file
```

Response. The process can use the information returned (if *sysmsg*[3] indicates no error occurred).

```
-109  NOWAIT FILENAME_FINDNEXT_ COMPLETION
```

Cause. The receiving process called FILENAME_FINDNEXT_ on a *searchid* that had a nowait operation specified for it by FILENAME_FINDSTART_. The results of the find request are being reported.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -109
sysmsg[1]          = The searchid on which the request was
                     made
sysmsg[2]          = The file-system error code giving the
                     resultant status of the request
sysmsg[3] FOR 5    = The type information (device type, and
                     so on) having the same layout as
                     described for the entityinfo parameter
                     of FILENAME_FINDNEXT_.  If sysmsg[2]
                     <> 0 the value of this field is
                     undefined.
sysmsg[8]          = Length in bytes of the returned name
                     (zero if none present because of an
                     error)
sysmsg[9] FOR 2    = The tag parameter from
                     FILENAME_FINDNEXT_
sysmsg[11] FOR 3   = Reserved, subject to change
sysmsg[14] FOR *   = The returned name, in external form
                     with a length given in sysmsg[8]
```

Response. The process can use the information returned (if *sysmsg*[2] indicates no error occurred).

```
-110  LOSS OF COMMUNICATION WITH NETWORK NODE
```

Cause. The remote node has either gone down or become partitioned from this node, and the process had called MONITORNET to enable reception of remote status change messages.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -110
sysmsg[1] FOR 3    = Reserved, subject to change
sysmsg[4] FOR 2    = Node identifier
sysmsg[6]          = Length of node name, in bytes
sysmsg[7] FOR n    = Node name (including the \)
```

Response. The response, if any, is application dependent.

```
-111  ESTABLISHMENT OF COMMUNICATION WITH NETWORK NODE
```

Cause. The remote node has established connection with this node, and the process had called MONITORNET to enable reception of remote status change messages.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -111
sysmsg[1] FOR 3    = Reserved, subject to change
sysmsg[4] FOR 2    = Node identifier
sysmsg[6]          = Length of node name, in bytes
sysmsg[7] FOR n    = Node name (including the \)
```

Response. The response, if any, is application-dependent.

```
-112  JOB PROCESS CREATION
```

Cause. The receiving process is the supervisor of a job and a process running under the supervisor's GMOM job ID created a third process.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -112
sysmsg[1]          = Job ID
sysmsg[2] FOR 10   = Process handle of the newly created
                     process
```

Response. The response, if any, is application-dependent.

```
-113  REMOTE PROCESSOR UP
```

Cause. The remote operating system has reloaded a processor, and the process had called MONITORNET to monitor remote status changes.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -113
sysmsg[1] FOR 2    = Node identifier
sysmsg[3]          = Processor number
sysmsg[4]          = Length of node name, in bytes
sysmsg[5] FOR 3    = Reserved
sysmsg[8] FOR *    = Node name (including the \)
```

Response. The response, if any, is application-dependent.

```
-121  PATHSEND DIALOG ABORT
```

Cause. A Pathsend dialog has been aborted for one of the following reasons:

- The requester aborted the dialog explicitly by calling the SERVERCLASS_DIALOG_ABORT_ procedure.
- The requester process abended.

- The requester canceled the last server-class send operation in the dialog.

Format. The operating system sends the server process the following message, provided the server has opened its $RECEIVE file to receive file-management system messages either by setting the FILE_OPEN_ procedure parameter *options*.<15> to 0 or by setting the OPEN procedure parameter *flags*.<1> to 1:

```
sysmsg [0] = -121
```

Response. The server process should reply to this system message with an error value of either FEOK (0) or FEEOF (1); these values direct the LINKMON process to release the link for re-use. To identify the dialog to which this message applies, the server must first call FILE_GETRECEIVEINFO_ to obtain the file number and process handle associated with this message. For more information about Pathsend dialogs, refer to the *NonStop TS/MP Pathsend and Server Programming Manual*.

```
-141  NOWAIT PROCESS_SPAWN_ COMPLETION
```

Cause. A call to the PROCESS_SPAWN_ procedure was completed.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]         = -141
sysmsg[1] FOR 13  = Reserved
sysmsg[14] FOR 2  = nowait-tag supplied to PROCESS_SPAWN_
sysmsg[16] FOR 2  = ZSYS-DDL-PROCESSRESULTS.Z-LEN of
                    process-results supplied to
                    PROCESS_SPAWN_
sysmsg[18] FOR 10 = ZSYS-DDL-PROCESSRESULTS.Z-PHANDLE of
                    process-results (process handle of
                    the new process)
sysmsg[28] FOR 2  = ZSYS-DDL-PROCESSRESULTS.Z-PID of
                    process-results (OSS pid of the
                    new process)
sysmsg[30] FOR 2  = ZSYS-DDL-PROCESSRESULTS.Z-ERRNO of
                    process-results (OSS errno)
sysmsg[32]        = ZSYS-DDL-PROCESSRESULTS.Z-TPCERROR of
                    process-results (Guardian error)
sysmsg[33]        = ZSYS-DDL-PROCESSRESULTS.Z-TPCDETAIL of
                    process-results (Guardian error
                    detail)
```

If this message returns an error value indicating that process creation failed, then it does not return a valid process handle or OSS process ID.

Response. The response, if any, is application-dependent.

```
 -147  DEVICE INFORMATION INQUIRY
```

Cause. The receiving process (which must have subtype 30 to receive this message) was the subject of a call to CONFIG_GETINFO_BYLDEV_, CONFIG_GETINFO_BYNAME_, or a similar procedure requesting physical device information. This message is used only on G-series releases.

Format. The layout of the received message described in terms of an array of words is:

```
sysmsg[0]          = -147
sysmsg[1]          = Message version (must be equal to
                     ZSYS-VAL-SMSG-CONFIGINFO-VERS)
sysmsg[2] FOR 4    = Device name for which configuration info
                     is needed; blank-filled (can be all blanks)
sysmsg[6] FOR 4    = Subdevice name; blank-filled (can be all
                     blanks)
sysmsg[10] FOR 4   = Secondary subdevice name qualifier;
                     blank-filled (can be all blanks)
```

Response. The receiver responds to system message -147 as follows:

If the receiver...	it calls REPLY with file-system error...
Does not handle message -147	2 (operation not allowed)
Does not recognize the version of the received message	565 (malformed request denied)
Does not recognize the device name or one of its qualifiers	14 (no such device)

If none of the preceding conditions apply, the receiver returns the requested information by calling REPLY with file-system error 0 (operation successful) and with data placed in the reply data buffer in the following form:

```
repmsg[0]          = -147
repmsg[1]          = Message version (must be equal to
                     ZSYS-VAL-SMSG-CONFIGINFO-VERS)
repmsg[2] FOR 2    = Device type; 32-bit value (can be -1D
                     for null)
repmsg[4]          = Device subtype (can be -1 for null)
repmsg[5]          = Device record size (must contain a
                     positive value)
repmsg[6]          = Logical status; the possible values are
                     defined in the file ZCOMDDL.
repmsg[7]          = Length in bytes of symbolic name of
                     physical hardware excluding the trailing
                     null (can be from 1 to 63)
repmsg[8] FOR *    = Symbolic name of physical hardware (null
                     terminated).  This is the same name that
                     is held in the CONFIG file and is the same
                     as the SCF object name.
repmsg[40]         = Length in bytes of subsystem manager name
```

```
                    excluding the trailing null (can be from 0
                    to 47)
repmsg[41] FOR *  = Subsystem manager name (null terminated);
                    can be either a simple name (for example,
                    $PDEV1) or a complete process descriptor
repmsg[65] FOR 10 = Process handle for primary I/O process;
                    can be null (that is, all words set to -1)
repmsg[75] FOR 10 = Process handle for secondary I/O process;
                    can be null (that is, all words set to -1)
repmsg[85] FOR 10 = Reserved for future use
repmsg[95]        = Length in bytes of device-specific
                    information; must be greater than or equal
                    to 0 and less than 2048
repmsg[96] FOR *  = Device-specific information; as they are
                    made available, structures defined by the
                    various subsystems are described in the
                    appropriate subsystem manuals.
```

Note. When evaluating the size of the data structure that contains everything preceding the device-specific information (as declared in ZSYS* files), TAL and pTAL give a different result from C. For example, if this data is declared as a structure called `data`, the correct length in TAL and pTAL would be `$len(data)-1`; the correct length in C would be `sizeof(data)-2`.

21 Traps and Signals

Certain critical problems can cause a process to be unable to continue executing. These are typically the result of coding errors, but other conditions, such as the lack of a system resource (for example, memory), can also prevent normal process execution. Such conditions are reported as traps to TNS processes and as signals to TNS/R native processes. A trap is a software mechanism that stops process execution. A signal is a software interrupt that can notify a process of other events, such as timer expiration, as well as of critical error conditions.

The set of signals that are used in the Guardian environment is known as TNS/R native signals. This set is a subset of a larger set of signals used in the Open System Services (OSS) environment. Most of the TNS/R native signals are caused by the same conditions that cause traps to occur in TNS processes. TNS processes in the Guardian environment do not receive signals, except under rare circumstances that cause a trap 8 to be generated.

In this section, equivalent trap and signal conditions are described together. Signals are commonly referred to by name, where each name is a literal that stands for a 32-bit signal number.

Table 21-1 lists the TNS/R native signals by name, in alphabetic order, along with their signal numbers, the equivalent trap numbers, and brief trap descriptions.

Table 21-1. TNS/R Native Signal Names, Signal Numbers, Trap Numbers, and Trap Descriptions

Signal Name	Signal Number	Trap Number	Trap Description
SIGABRT	6D	none	(not available to TNS processes)
SIGALRM	14D	none	(not available to TNS processes)
SIGFPE	8D	2	Arithmetic overflow
SIGILL	4D	1	Instruction failure
SIGLIMIT	27D	5	Limits exceeded
SIGMEMERR	22D	13	Uncorrectable memory error
SIGMEMMGR	24D	11	Memory manager read error
SIGNOMEM	23D	12	No memory available
SIGSEGV	11D	0	Illegal address reference
SIGSTK	25D	3	Stack overflow
SIGTIMEOUT	26D	4	Process loop-timer timeout
other signals	--	8	(signal delivered to TNS process)

Trap Handling

When a trap occurs, control passes to the Debug or Inspect debugger by default. However, by using the ARMTRAP system procedure, a program can specify a trap handler to be executed when the trap occurs or, alternatively, that a trap causes the program to abend. The actions of a trap handler typically include either terminating the process or, in certain cases, resuming execution.

For information about the Debug and Inspect debuggers, refer to the *Debug Manual* and the *Inspect Manual*.

For a description of the ARMTRAP procedure, see the *Guardian Procedure Calls Reference Manual*. For an explanation of how to create your own trap handler with ARMTRAP, see the *Guardian Programmer's Guide*.

Signal Handling

When a process receives a TNS/R native signal, the default action is for the process to abend. Alternatively, the program can call the SIGACTION_INIT_ procedure to specify that another action should result, such as the execution of a signal handler. If a signal handler has been specified, it is executed when the process receives a signal. The actions of a signal handler typically include either terminating the process or, in certain cases, resuming execution.

For additional information about signals and signal handling, refer to the *Guardian Programmer's Guide.* For a description of the SIGACTION_INIT_ procedure, see the *Guardian Procedure Calls Reference Manual*.

Signals and Trap Numbers

This subsection lists each signal by name with its equivalent trap number and provides a description of each. In general, it includes only signals that are generated by the system; additional signals can be generated by the `raise()` function. The list is in order by trap number. Signals that have no equivalent trap are listed at the end of this subsection.

```
SIGSEGV, Trap 0  Illegal address reference
```

Cause. An address was specified that was not within either the virtual code area or the virtual data area allocated to the process.

Note. If a TNS process has fewer than 32 pages of data stack, then a situation such as an infinitely recursive TNS procedure causes a trap 0 instead of a trap 3.

References to privileged 32-bit addresses from unprivileged code are reported as SIGSEGV or trap 0.

Effect. For trap 0, process control passes to the Debug or Inspect debugger, or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGSEGV, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. If the illegal address reference occurred because of an erroneous pointer value, correct the program. For TNS processes, if the address is lower than 65535. refer to the recovery information for trap 3.

```
SIGILL, Trap 1  Instruction failure
```

Cause. The instruction failure is caused by one of the following:

- There was an attempt to execute a code word that is not a valid instruction.
- A nonprivileged process tried to either execute a privileged instruction or access a privileged memory address.
- An EXTENSIBLE procedure was called with a calling sequence that involved a mismatch. Most commonly, the mismatch is the following: The calling program was compiled using an external declaration that specified more parameters than were declared for the procedure being called, and the calling program passed one of these unsupported parameters.
- The program has executed a deliberate failure instruction emitted by the compiler to cause a trap when an error situation was detected at run time. (For example, a pTAL program executed an END statement without encountering a RETURN statement.)

Effect. For trap 1, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGILL, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. Check your application program.

```
SIGFPE, Trap 2  Arithmetic overflow
```

Cause. Overflow trapping is enabled, and an arithmetic overflow occurred. In the TNS environment, overflow traps are enabled when the T bit (ENV.<8>) is set to 1. In TNS/R native mode, overflow checking is controlled statically at compile time. Arithmetic overflow occurs for one of the following reasons:

- The result of a signed arithmetic operation could not be represented with the number of bits available for the particular data type.
- A division operation with a divisor of zero was attempted, or unsigned integer division resulted in a quotient that exceeded the word size.
- The control variable of a CASE expression, or a CASE statement with no OTHERWISE label, did not match any of the defined cases.

Effect. For trap 2, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGFPE, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. If the overflow occurred because of an erroneous computation, correct the program.

If overflow is legitimate at this point, either use unsigned arithmetic (for 16-bit values) or disable overflow traps during the computation. One example of when overflow is sometimes legitimate is an address computation.

For trap 2, the program can use a trap handler to detect the overflow and recover. The recovery action can be either to resume the computation or to transfer control to a recovery point. However, resuming the computation at the point of the trap is not an option if the trap occurred in an operating-system procedure.

Note that before returning to the program at the trap point or elsewhere, the trap handler must clear the V bit (bit 10) in the trap handler’s copy of the ENV register at L[-1]. Otherwise, the process will abend.

A signal handler cannot resume the process at the site of the overflow.

```
SIGSTK, Trap 3  Stack overflow
```

Cause. The cause of the stack overflow varies with the type of process.

- In TNS or accelerated mode, a stack overflow results when one of the following problems occurs:
    - The value in the TNS stack register, S, exceeds 32767. Typically, an attempt was made to execute a procedure or subprocedure whose stack marker, sublocal data area, or local data area extends into the upper 32K of the user data segment.
    - There was not enough remaining virtual data space for an operating-system procedure to execute.

  The amount of virtual data space available is the lesser of 'G'[32767] and the upper bound of the process's virtual data area (the number of data pages specified when the process was created or run).

  Operating-system procedures require approximately 350 words of user-data stack space to execute.

Note

If you have fewer than 32 pages of data stack, then a situation such as an infinitely recursive TNS procedure causes a trap 0 instead of a trap 3.

- In TNS/R native mode, the cause of the stack overflow is as follows:

  A process running in TNS/R native mode has set the stack pointer register (SP) to point beyond the allocated RISC stack but within the same memory region, and the process has attempted to access a memory address between the end of the stack and the address designated by the SP. Typically, this situation occurs when a procedure or subprocedure was called with its activation record extending beyond the end of the RISC stack.

Effect. For trap 3, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGSTK, the process abends or enters the Debug or Inspect debugger. A SIGSTK signal cannot be delivered to a signal handler because there is no space on the stack for invoking the handler. See Signal Handling on page 21-2 for further information.

Recovery. Recovery is application dependent. If the problem is infinite recursion of procedure calls, correct the program.

For TNS processes, you can increase the size of the virtual data area with one of the following:

- The TAL compiler DATAPAGES directive
- The MEM parameter of the command interpreter RUN command
- The *memory-pages* parameter of either the NEWPROCESS or PROCESS_CREATE_ procedure, the Z^MEMORY^PAGES field of the *param-list* parameter of the PROCESS_LAUNCH_ procedure, or the Z^MEMORYPAGES field of the *process-extension* parameter of the PROCESS_SPAWN_ procedure.

If there is simply too much data for the TNS stack segment, you can revise the program by putting some of its global or local data into an extended data segment.

For TNS/R processes, the main stack grows dynamically as needed, up to a maximum. You can increase the maximum when the process is created, up to a limit.

For further information about managing memory, refer to the *Guardian Programmer's Guide.*

```
SIGTIMEOUT, Trap 4  Process loop-timer timeout
```

Cause. The time limit specified in the latest call to SETLOOPTIMER has expired.

Effect. For trap 4, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGTIMEOUT, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Note that the loop-timer trap or signal is disabled after the trap or signal is generated; the loop-timer trap or signal will not recur unless it is rearmed by another call to SETLOOPTIMER.

Recovery. Recovery is application dependent. It is possible to resume from both trap 4 and SIGTIMEOUT, continuing the program from the site of the interruption.

```
SIGLIMIT, Trap 5  Limits exceeded
```

Cause. This condition occurs for one of the following reasons:

- The process called an operating-system procedure that tried to place a process identification number (PIN) greater than 255 into a 1-byte field.
- A TNS Guardian process invoked a procedure or function that is allowed only for an OSS or TNS/R native process.

Effect. For trap 5, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGLIMIT, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. Run the process at a low PIN or recode the process.

```
Trap 8  (Signal delivered to TNS process)
```

Cause. A trap 8 is generated when, under very rare circumstances, a signal is delivered to a TNS process.

Effect. Process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

Recovery. Contact your system manager.

```
SIGMEMMGR, Trap 11  Memory manager read error
```

Cause. A hard (unrecoverable) read error occurred while the program was trying to bring a page in from virtual memory.

Effect. For trap 11, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGMEMMGR, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

If the trap or signal occurred because the disk containing the swap file went down, the process abends, regardless of the selected trap or signal handling.

Recovery. Recovery is application dependent. Usually, this condition occurs when both the primary and backup paths to a disk are down. Contact your system manager to resolve the problem.

```
SIGNOMEM, Trap 12  No memory available
```

Cause. This error occurs for one of the following reasons:

- A page fault occurred, but no physical memory page is available for overlay.
- Disk space cannot be allocated when the program is referencing an extensible segment.

Effect. For trap 12, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGNOMEM, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. If the program was using extensible segments, either free up disk space on the swap volume or specify a swap file on a different volume.

```
SIGMEMERR, Trap 13  Uncorrectable memory error
```

Cause. An uncorrectable memory error occurred.

Effect. For trap 13, process control passes to the Debug or Inspect debugger or to the trap handler designated by a call to the ARMTRAP procedure, or else the process is abended. See Trap Handling on page 21-2 for further information.

For the signal SIGMEMERR, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. If possible, rerun your application. Otherwise, contact your system manager.

```
SIGABRT  (No trap)  Abnormal termination
```

Cause. A procedure running within the process called `abort()` or `raise(SIGABRT)`.

Effect. For the signal SIGABRT, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. Recovery is application dependent.

```
SIGALRM  (No trap)  Timer expiration
```

Cause. A procedure called `alarm()` and the specified time has elapsed.

Effect. For the signal SIGALRM, the default or specified signal handling occurs. See Signal Handling on page 21-2 for further information.

Recovery. Recovery is application dependent. It is possible to resume from SIGALRM and to continue the program from the site of the interruption.

Error Lists

If you are using the Subsystem Programmatic Interface (SPI) to send commands to a subsystem, you might receive a trap-out error list in a response. HP subsystems return such an error list when, in performing your request, they encounter either a trap condition or an exception condition that causes a signal to be generated.

Each error list always includes the unconditional tokens listed under its description. If you are designing a subsystem that uses SPI, include all unconditional tokens listed in the error list's description.

This subsection does not discuss the mechanics of error-list construction. For information about creating error lists, for additional information about tokens and token types, and for definitions of tokens whose names begin with ZSPI-, refer to the *SPI Programming Manual*.

5: ZGRD-VAL-TRAPOUT

A user-enabled trap or signal handler started execution following the detection of a trap or exception condition while a process was executing.

Unconditional Tokens

```
ZSPI-TKN-ERRLIST                token-type ZSPI-TYP-LIST.
   ZSPI-TKN-ERROR               token-type ZSPI-TYP-ERROR.
   ZSPI-TKN-PROC-ERR            token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ENDLIST                token-type ZSPI-TYP-SSCTL.
```

Conditional Tokens

```
   ZGRD-TKN-XOBJECTFILE         token-type ZSPI-TYP-STRING.
   ZGRD-TKN-ENTRYPOINTLABEL     token-type ZSPI-TYP-STRING.
   ZGRD-TKN-BINDTIMESTAMP       token-type ZSPI-TYP-TIMESTAMP.
   ZGRD-TKN-STACKENV            token-type ZSPI-TYP-UINT.
   ZGRD-TKN-SREGISTER           token-type ZSPI-TYP-UINT.
   ZGRD-TKN-PREGISTER           token-type ZSPI-TYP-UINT.
   ZGRD-TKN-EREGISTER           token-type ZSPI-TYP-UINT.
   ZGRD-TKN-LREGISTER           token-type ZSPI-TYP-UINT.
   ZGRD-TKN-SIGNAL-NUM          token-type ZSPI-TYP-INT2.
   ZGRD-TKN-TRAP-TEXT           token-type ZSPI-TYP-STRING.
```

Unconditional Tokens

ZSPI-TKN-ERROR is the standard SPI error token, whose value consists of the fields Z-SSID and Z-ERROR. Z-SSID is the subsystem identifier ZGRD-VAL-SSID. Z-ERROR is the trap or signal number.

ZSPI-TKN-PROC-ERR is the procedure code. Its value is ZGRD-VAL-TRAPOUT (5).

Conditional Tokens

ZGRD-TKN-XOBJECTFILE is the name of the program file associated with the process that encountered the trap or signal.

ZGRD-TKN-ENTRYPOINTLABEL is the internal procedure entrypoint label of the trapping procedure (IPIL entry).

ZGRD-TKN-BINDTIMESTAMP is the program file's `nld` utility or Binder timestamp.

ZGRD-TKN-STACKENV is the value of the stack-marker ENV register at the time the trap occurred. This token is applicable only to TNS processes.

ZGRD-TKN-SREGISTER is the value of the S register at the time of the trap. This token is applicable only to TNS processes.

ZGRD-TKN-PREGISTER is the value of the P register at the time of the trap. This token is applicable only to TNS processes.

ZGRD-TKN-EREGISTER is the value of the E register at the time of the trap. This token is applicable only to TNS processes.

ZGRD-TKN-LREGISTER is the value of the L register at the time of the trap. This token is applicable only to TNS processes.

ZGRD-TKN-SIGNAL-NUM is the signal number of the signal that was received. This token is applicable only to TNS/R native processes.

ZGRD-TKN-TRAP-TEXT is a text representation of the process state. It is recommended that your program use the HIST_INIT_ and HIST_FORMAT_ procedures to produce the text, as these procedures are capable of handling and formatting the process state of both TNS and TNS/R native processes. To produce the most concise text, specify the options HO_Init_uContext and HO_OneLine in the call to HIST_INIT_ and specify the options HF_base and HF_LocLineTNS (or HF_LocLineRISC, as appropriate) in the call to HIST_FORMAT_. For further information about the HIST_INIT_ and HIST_FORMAT_ procedures, refer to the *Guardian Procedure Calls Reference Manual*.

Effect

The process might not be able to continue executing.

Recovery

Take corrective action as indicated by the returned trap or signal number. Traps and signals are described earlier in this section.

22 OSS Error Information

This section describes how to find more information on Open System Services (OSS) errors.

Brief OSS Error Information

Guardian procedures that make use of OSS functions can return OSS `errno` values in addition to Guardian file-system error codes.

You can obtain a short explanation of any OSS `errno` value or Guardian file-system error code at the TACL prompt by entering:

```
ERROR number
```

If you want to scan a list of all the errors, enter:

```
ERROR -1
```

Detailed OSS Error Information

Obtaining a more detailed explanation of an OSS error requires two steps:

1. In the `errnoh` file, look up the OSS `errno` symbolic name that corresponds to the OSS `errno` value returned.
2. To find the error description, look up the OSS `errno` symbolic name in the *Open System Services System Calls Reference Manual*.

Example

For example, to find a description for OSS `errno` value 4028 from the Guardian environment, perform the following steps:

1. Using an editor, search the `errnoh` file for the OSS `errno` symbolic name corresponding to 4028:

   ```
   > edit $system.system.errnoh read
   TEXT EDITOR - T9601B30 - (08MAR87)
   CURRENT FILE IS \MYSYS.$SYSTEM.SYSTEM.ERRNOH
   *list unseq "4028"
   #define ENOSPC    4028  /* No space left on device   */
   ...
   *exit
   >
   ```

   The OSS `errno` symbolic name corresponding to 4028 is `ENOSPC`.

2. Refer to the *Open System Services System Calls Reference Manual* to find the description of `ENOSPC`:

   [ENOSPC] No space left on device. During the **write( )** function on a regular file or when extending a directory, there is no free space left on the device.

Index