File Utility Program (FUP) Management Programming Manual

Abstract

This manual describes the programmatic interface for the File Utility Program (FUP) and the Online Reload Server (ORSERV). It is intended for programmers who are writing applications that communicate with FUP or ORSERV.

Product Version

Utilities D46

Supported Releases

This manual supports G06.00, D46.00, and all subsequent G0x.xx and D4x.xx releases until otherwise indicated in a new edition.

Part Number	Published
523322-001	February 2002

Document History

Part Number	Product Version	Published
098213	Utilities D20	September 1993
136591	Utilities D46	April 1998
425743-001	Utilites D46	August 2000
429477-001	Utilities D46	July 2001
523322-001	Utilities D46	February 2002

File Utility Program (FUP) Management Programming Manual

Index | Figures | Tables

2. FUP Programmatic Interface (continued)

3. FUP Commands and Responses

4. ORSERV Programmatic Interface

5. ORSERV Commands and Responses

5. ORSERV Commands and Responses (continued)

A. Management Application Example

B. FUP Error Messages

B. FUP Error Messages (continued)

B. FUP Error Messages (continued)

C. ORSERV Error Messages

Index

Figures

Figures (continued)

Tables

What's New in This Manual

Manual Information

Abstract

This manual describes the programmatic interface for the File Utility Program (FUP) and the Online Reload Server (ORSERV). It is intended for programmers who are writing applications that communicate with FUP or ORSERV.

Product Version

Utilities D46

Supported Releases

This manual supports G06.00, D46.00, and all subsequent G0x.xx and D4x.xx releases until otherwise indicated in a new edition.

Part Number	Published
523322-001	February 2002

Document History

Part Number	Product Version	Published
098213	Utilities D20	September 1993
136591	Utilities D46	April 1998
425743-001	Utilites D46	August 2000
429477-001	Utilities D46	July 2001
523322-001	Utilities D46	February 2002

New and Changed Information

This edition of the *File Utility Program (FUP) Management Programming Manual* contains these changes:

- The sample code in Appendix A, Management Application Example, was optimized.

About This Manual

Who Should Read This Manual?

This manual describes the programmatic interface for the File Utility Program (FUP) and its Online Reload Server (ORSERV) process. The Subsystem Programmatic Interface (SPI) and its message-based, token-oriented interface let programmers write management applications that can communicate with the FUP subsystem.

Any C, COBOL85, or TAL language programmers who are developing a management application that uses SPI to communicate with FUP or ORSERV should use this manual. It might also benefit Compaq Tandem Advanced Command Languague (TACL) programmers who are writing macros or routines that use SPI to communicate with FUP or ORSERV, and any others who need detailed information about the FUP or ORSERV programmatic interface.

Readers of this manual should be familiar with the Compaq *NonStop*™ Kernel operating system and its file-system terminology. You should also be familiar with using interactive FUP commands, the Compaq requester-server approach to application programming, the Compaq Data Definition Language (DDL), and SPI and one of its compatible programming languages (C, COBOL85, TACL, or TAL).

Organization of This Manual

Section/Appendix	Description
Section 1, Introduction	Introduces the programmatic interface used by FUP and ORSERV, describes the FUP structure and its relationship to ORSERV, and displays the differences of using interactive FUP commands, programmatic commands, or file-system procedures on C-series or D-series systems
Section 2, FUP Programmatic Interface	Describes how a management application communicates with FUP using SPI
Section 3, FUP Commands and Responses	Details the FUP CHECKSUM, DUPLICATE, GETVERSION, LOAD, LOADALTFILE, and RESTART programmatic commands
Section 4, ORSERV Programmatic Interface	Describes how a management application communicates with ORSERV using SPI
Section 5, ORSERV Commands and Responses	Details the ORSERV GETVERSION, ONLINERELOAD, STATUS, and SUSPEND programmatic commands

Section/Appendix	Description
Appendix A, Management Application Example	Displays an example listing output using the COBOL85 management application to communicate with FUP and execute the programmatic DUPLICATE command
Appendix B, FUP Error Messages	Describes the FUP errors and supplies a recommended course of action to correct or alleviate each one of them
Appendix C, ORSERV Error Messages	Describes the ORSERV errors and supplies a recommended course of action to correct or alleviate each one of them

Related Reading

Compaq recommends these manuals as references:

- *Data Definition Language (DDL) Reference Manual*
- *Introduction to Distributed Systems Management (DSM)*
- *File Utility Program (FUP) Reference Manual*
- *Guardian Procedure Calls Reference Manual*
- *Guardian Procedure Errors and Messages Manual*

Your Comments Invited

After using this manual, please take a moment to send us your comments. You can do this by:

- Completing a Contact *NonStop™ Himalaya* Publications form online at http://nonstop.compaq.com/view.asp?FOID=20.
- Faxing or mailing the form, which is included as a separate file in Total Information Manager (TIM) collections and located at the back of printed manuals. Our fax number and mailing address are included on the form.
- Sending an e-mail message to the address included on the form. We'll immediately acknowledge receipt of your message and send you a detailed response as soon as possible. Be sure to include your name, company name, address, and phone number in your message. If your comments are specific to a particular manual, also include the part number and title of the manual.

Many of the improvements you see in manuals are a result of suggestions from our customers. Please take this opportunity to help us improve future manuals.

Notation Conventions

Hypertext Links

Blue underline is used to indicate a hypertext link within text. By clicking a passage of text with a blue underline, you are taken to the location described. For example:

> This requirement is described under Backup DAM Volumes and Physical Disk Drives on page 3-2.

General Syntax Notation

This 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; enter 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 this 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 ]...
```

!i and !o. In procedure calls, the !i notation follows an input parameter (one that passes data to the called procedure); the !o notation follows an output parameter (one that returns data to the calling program). For example:

```
CALL CHECKRESIZESEGMENT (  segment-id                    !i
                         , error        ) ;              !o
```

!i,o. In procedure calls, the !i,o notation follows an input/output parameter (one that both passes data to the called procedure and returns data to the calling program). For example:

```
error := COMPRESSEDIT ( filenum ) ;                      !i,o
```

!i:i. In procedure calls, the !i:i notation follows an input string parameter that has a corresponding parameter specifying the length of the string in bytes. For example:

```
error := FILENAME_COMPARE_ (  filename1:length           !i:i
                            , filename2:length ) ;       !i:i
```

!o:i. In procedure calls, the !o:i notation follows an output buffer parameter that has a corresponding input parameter specifying the maximum length of the output buffer in bytes. For example:

```
error := FILE_GETINFO_ (  filenum                        !i
                        , [ filename:maxlen ] ) ;        !o:i
```

Notation for Messages

This list summarizes the notation conventions for the presentation of displayed messages in this manual:

Bold Text. Bold text in an example indicates user input entered at the terminal. For example:

```
ENTER RUN CODE
?123
CODE RECEIVED:      123.00
```

The user must press the Return key after typing the input.

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

This list summarizes the notation conventions used in the boxed descriptions of programmatic commands, event messages, and 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
```

!r. The !r notation following a token or field name indicates that the token or field is required. For example:

```
ZCOM-TKN-OBJNAME        token-type ZSPI-TYP-STRING.          !r
```

!o. The !o notation following a token or field name indicates that the token or field is optional. For example:

```
ZSPI-TKN-MANAGER        token-type ZSPI-TYP-FNAME32.         !o
```

1 Introduction

You use the File Utility Program (FUP) and its Online Reload Server (ORSERV) process to create and manage disk files. Applications can use the Subsystem Programmatic Interface (SPI) to communicate with FUP or its ORSERV process.

Note. This manual describes SPI information that is specific to FUP and ORSERV. For more information about SPI, see the *SPI Programming Manual*.

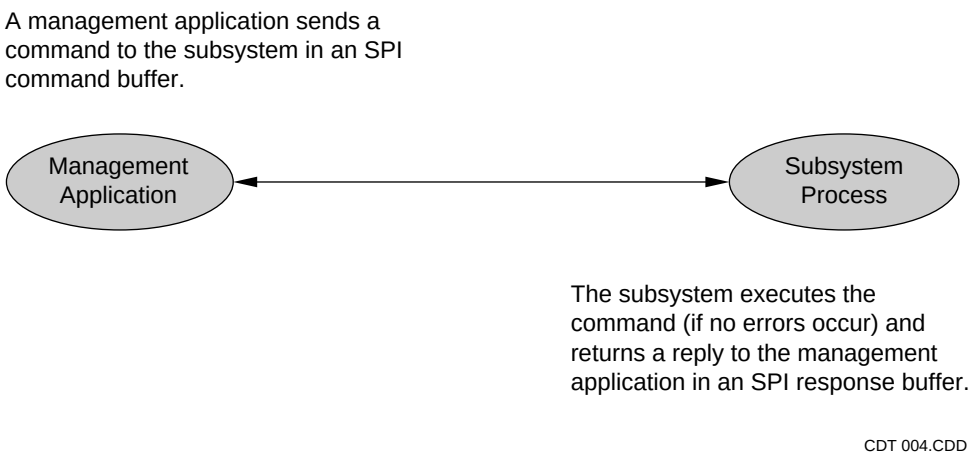
Subsystem Programmatic Interface (SPI) Overview

The Subsystem Programmatic Interface (SPI) is a message-based interface you can use to build and decode messages that are needed to communicate between requesters and servers. It lets you write a management application (requester) in COBOL85, C, TACL, or TAL to communicate with a subsystem (server) such as FUP.

Note. For an example of a management application (COBOL85), see Appendix A, Management Application Example.

Your management application uses the SPI procedures to create a specialized, formatted message that is identified as the command and response buffer. You can send and receive this buffer by using any of the NonStop Kernel interprocess communication methods—including the TAL WRITEREAD message procedure. The information in the buffer consists of tokens that are identified by symbolic names instead of buffer addresses or positions. Figure 1-1 shows an example of a management application communicating with a subsystem process.

Figure 1-1. A Management Application and a Subsystem Process



FUP Subsystem Overview

The FUP subsystem consists of the FUP and ORSERV processes. The FUP process creates and manages disk files, and the ORSERV process reloads key-sequenced Enscribe files and SQL objects. You can access FUP and ORSERV interactively from a terminal, or you can access them programmatically from a management application written in COBOL85, C, TACL, or TAL.

When you enter a FUP command interactively, you create a FUP process by entering the FUP command at the TACL prompt (which is actually an implicit form of the TACL RUN command). You then enter FUP commands that are interpreted and executed by the FUP process.

Note. For more information about the FUP commands, see the *File Utility Program (FUP) Reference Manual*.

Although most of the FUP interactive commands are executed by the FUP process, FUP creates the ORSERV process to handle these commands:

- RELOAD
- STATUS
- SUSPEND

RELOAD

RELOAD performs an online reload of a key-sequenced file or SQL object. Key-sequenced files that have had several insertions, deletions, or updates with key-length changes can have the following abnormalities:

- Empty blocks or blocks containing large amounts of empty space that cause an inefficient use of disk space and slower sequential access time

- An increased number of index levels that cause slower random access time
- Physical declustering of data on the disk causing slower sequential access time (due to an increased amount of disk read/write head movement), and a reduced capability to perform large I/O transfers

An online reload operation performed by ORSERV eliminates these abnormalities. During the reload operation, ORSERV allows the shared-exclusion mode and the read/write access mode for the target file.

STATUS

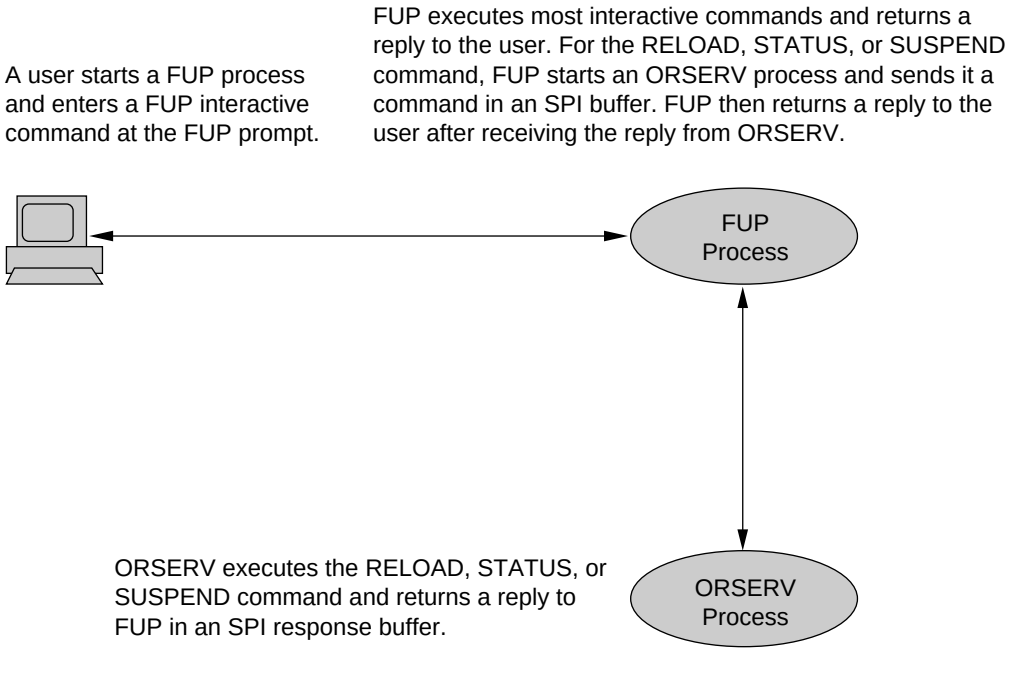
STATUS returns the status of a reload operation for either an executing or a completed reload operation. Use this command to monitor the progress of a reload operation and determine if it should be suspended.

SUSPEND

SUSPEND suspends an executing online reload operation. Use this command to stop an ORSERV process that is performing a reload operation. This command lets you suspend an ORSERV process (and restart it later) if reload operations are degrading the performance of other applications on the system.

The ORSERV process is created automatically when you use FUP (interactively) to execute the RELOAD, STATUS, or SUSPEND commands. Figure 1-2 shows an example of FUP creating the ORSERV process.

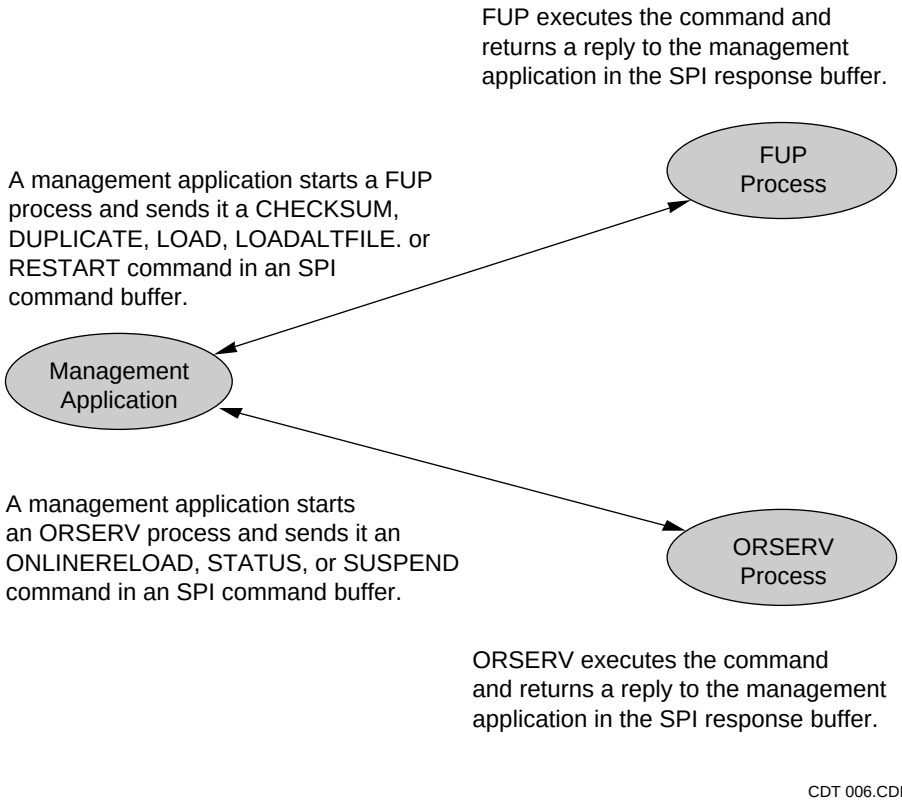
Figure 1-2. Using FUP Interactively



Using SPI With FUP and ORSERV

Your management application (COBOL85, C, TACL, or TAL) communicates with FUP and ORSERV processes using SPI. These two programmatic interfaces (FUP and ORSERV) are used to execute specific commands. Figure 1-3 shows an example of a management application communicating with both a FUP and ORSERV process.

Figure 1-3. Management Application With FUP and ORSERV



SPI With FUP

You can use the FUP programmatic interface to execute the CHECKSUM, DUPLICATE, LOAD, LOADALTFILE, RESTART, and GETVERSION commands. Use your management application to create a FUP process, open the process, and then send your command requests to it by using an SPI command buffer. For more information, see Section 2, FUP Programmatic Interface.

SPI With ORSERV

You can use the ORSERV programmatic interface to execute the ONLINERELOAD, STATUS, SUSPEND, and GETVERSION commands. Use your management application to create an ORSERV process, open the process, and then send your command request to it.

Your management application sends commands directly to the ORSERV process. For more information, see Section 4, ORSERV Programmatic Interface.

ORSERV Management Programming

Although you can reload a key-sequenced file using the interactive FUP RELOAD command, it requires operator intervention. An operator must decide when to initiate the reload operation, when to suspend the reload by using the SUSPEND command (if necessary), and when to resume the reload with the RELOAD command.

You can write a management application that performs these duties automatically and eliminates the need for operator intervention. You can design your management application to initiate, suspend, and restart reload operations:

- Initiate the reload when it has minimal effect on other applications that are running on the system.
- Reload each partition of a partitioned file (that resides on different systems) when it has a minimal effect on each system.
- Monitor the execution of other applications running on the system and suspend the reload until it has minimal impact (or the other applications have completed).
- Generate status reports at specific intervals to show the progress of a reload operation. You can store these reports in a disk file for later reference or have them displayed at a specific terminal for immediate use.
- Restart the reload with the ZRATE field of ZORS-MAP-PAR-ONLINERELOAD set to a lower value.

Note. The ZRATE field determines the percentage of execution time that ORSERV spends performing the reload. The ORSERV process spends the remainder of its execution time in delay mode to let other applications use processor time, physical memory, and other system resources.

Executing a Sequence of ORSERV Commands

The management application you use must create an ORSERV process for each programmatic ORSERV command it executes:

1. Your management application creates an ORSERV process and sends it an ONLINERELOAD command (to perform an online reload operation for a target file).

   The ORSERV process initiates the reload operation, sends a reply to your management application in the SPI response buffer, and continues the reload before closing the ORSERV process.

2. Your management application executes a STATUS command (at a later, predetermined time) to discover the progress of the reload operation. It must create a second ORSERV process and send it a STATUS command that specifies the original target file.

   The second ORSERV process determines the status of the reload operation from the ZZRELOAD file and sends the status to your management application in the SPI response buffer.

   For more information about the ZZRELOAD file, see Section 4, ORSERV Programmatic Interface.

3. After your management application analyzes the reload status, it suspends the reload operation. It must create a third ORSERV process and send it a SUSPEND command that also specifies the original target file.

   This ORSERV process (Step 3) stops the first one and the information about the progress of the reload is saved in the ZZRELOAD file.

FUP Command and Procedure Differences

Table 1-1 shows the interactive FUP commands that you can execute with FUP or ORSERV programmatic commands. This table also provides the NonStop Kernel file-system procedure for executing the commands (if applicable) and distinguishes between the C-series and D-series file-system procedures. Programmatic commands were developed for most of the FUP functions that did not have a file-system procedure.

Note. Both C-series and D-series file-system procedure calls are supported on G-series systems.

Table 1-1. FUP Commands and File-System Procedures for C-Series and D-Series File Systems (page 1 of 5)

FUP Interactive Command	Programmatic Command	C-Series File-System Procedure	D-Series File-System Procedure	Description
	ZFUP-CMD-GET VERSION			Returns the FUP version and the server ID string Returns the ORSERV server version and the server ID string
ALLOCATE		CONTROL	CONTROL	Preallocates disk file extents
ALTER		ALTER	FILE_ ALTERLIST_	Changes the file code, alternate-key specifications, and partition specifications of a disk file Also changes the RESETBROKEN, REFRESH, AUDIT, and ODDUNSTR attributes of a disk file

Table 1-1. FUP Commands and File-System Procedures for C-Series and D-Series File Systems (page 2 of 5)

FUP Interactive Command	Programmatic Command	C-Series File-System Procedure	D-Series File-System Procedure	Description
ALTER		SETMODE	SETMODE	Changes the MAXEXTENTS, BUFFERSIZE, AUDITCOMPRESS, BUFFERED, SERIALWRITES, and VERIFIEDWRITES attributes of a disk file
				Recomputes the checksum value for blocks of data in disk files
CHECK SUM	ZFUP-CMD-CHECK SUM			Recomputes the checksum value for blocks of data in disk files
CREATE		CREATE	FILE_CREATE_	Creates a disk file
			FILE_CREATE_LIST_	
DUPLICATE	ZFUP-CMD-DUPLICATE			Makes a copy of one or more disk files, except when using the restartable option, which allows the copy of only one disk file
DEALLOCATE		CONTROL	CONTROL	Deallocates disk file extents
GIVE		SETMODE	SETMODE	Changes the owner of a disk file

Table 1-1. FUP Commands and File-System Procedures for C-Series and D-Series File Systems (page 3 of 5)

FUP Interactive Command	Programmatic Command	C-Series File-System Procedure	D-Series File-System Procedure	Description
INFO		FILREC INFO	FILE_ GETINFO_	Determines whether a file has an alternate key
			FILE_ GETINFO BY NAME_	Identifies record length, block length, index block length, key length, key offset, alternate-key specifications, alternate-key file specifications, and partition-file specifications
			FILE_ GETINFO LIST_	
			FILE_ GETINFO LIST BYNAME_	
INFO		FILEINFO	FILE_ GET INFO_	Identifies the open state, file code, end of file, modification (last written) date, owner security, and type for a disk file. Also identifies the MAXEXTENTS, EXT, BUFFERSIZE, REFRESH, BUFFERED, AUDITCOMPRESS, VERIFIEDWRITES, SERIALWRITES, DCOMPRESS, ICOMPRESS, AUDIT, ODDUNSTR, PROGID, CLEARONPURGE, LICENSE, EXTENTSALLOCATE, and CORRUPT attributes of a disk file
			FILE_ GETINFO BYNAME_	
			FILE_ GETINFO LIST_	
			FILE_ GETINFO LISTBY NAME_	

Table 1-1. FUP Commands and File-System Procedures for C-Series and D-Series File Systems (page 4 of 5)

FUP Interactive Command	Programmatic Command	C-Series File-System Procedure	D-Series File-System Procedure	Description
INFO		FILE INQUIRE	FILE_ GETINFO_ FILE_ GETINFO BY NAME_ FILE_GET INFOLIST_ FILE_GET INFOLISTBY NAME_	Identifies the number of levels of a key-sequenced file, generic lock length, time of creation, and time of last open
LICENSE		SETMODE	SETMODE	Lets a program file run privileged processes
LISTLOCKS		LOCKINFO	FILE_GET LOCK INFO_	Identifies file-lock information
LISTOPENS		OPENINFO	FILE_GET OPEN INFO_	Identifies the process that has a file open and the access mode and exclusion mode of an open operation
LOAD	ZFUP-CMD-LOAD			Loads data into a structured disk file without affecting any associated alternate-key files
LOAD ALTFILE	ZFUP-CMD-LOAD ALTFILE			Generates alternate-key records from a specified primary file and loads the alternate-key records into an alternate-key file
PURGE		PURGE	FILE_ PURGE_	Deletes a disk file
PURGEDATA		CONTROL	CONTROL	Removes data from a file
RELOAD				Performs an online reload of a key-sequenced file or SQL object

Table 1-1. FUP Commands and File-System Procedures for C-Series and D-Series File Systems (page 5 of 5)

FUP Interactive Command	Programmatic Command	C-Series File-System Procedure	D-Series File-System Procedure	Description
RENAME		RENAME	FILE_ RENAME_	Changes a file or volume name
RESTART	ZFUP-CMD-RESTART			Allows the continuation of a DUPLICATE command (with the restartable option) that failed
REVOKE		SETMODE	SETMODE	Revokes the license of a program to run privileged procedures
SECURE		SETMODE	SETMODE	Changes the NonStop Kernel security attributes of a disk file
STATUS				Returns the status of a reload operation (either executing or finished)
SUSPEND				Suspends an executing reload operation

2 FUP Programmatic Interface

This section describes how a management application program uses the Subsystem Programmatic Interface (SPI) to communicate with a FUP process.

You can use SPI to execute these FUP commands from an application written in COBOL85, C, TACL, or TAL:

CHECKSUM	Recomputes the checksum value for blocks of data in disk files
DUPLICATE	Makes a copy of one or more disk files unless the restartable option is being used (it only allows the duplication of one disk file)
GETVERSION	Returns the FUP server version and the server ID
LOAD	Loads data into a structured disk file without affecting any associated alternate-key files
LOADALTFILE	Generates alternate-key records from a specified primary file and loads the alternate-key records into an alternate-key file
RESTART	Allows the continuation of a DUPLICATE command (with the restartable option specified) that failed

Note. Before reading this section, you should be familiar with the *SPI Programming Manual.*

Communicating With FUP

To communicate with a FUP process:

1. Start a FUP process.

   You can use a process creation procedure, such as PROCESS_LAUNCH_. FUP is in the $SYSTEM.SYS*nn*.FUP program file (where *nn* is a two-digit octal integer that identifies the subvolume).

   Note. For more information on process creation, see the *Guardian Procedure Calls Reference Manual.*

2. Send FUP a startup message.

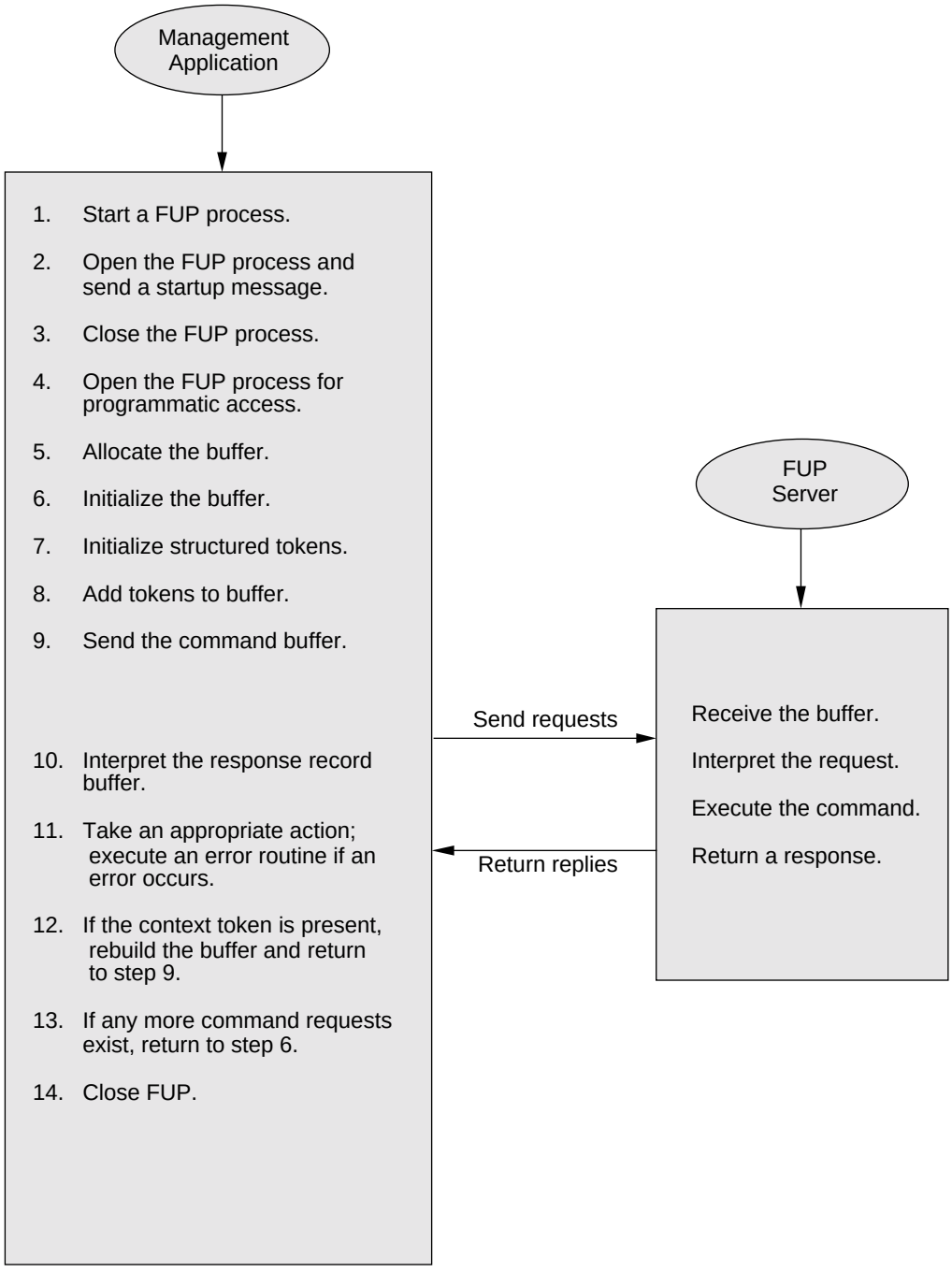
   Open the FUP process, then send FUP a startup message using these guidelines:

   - The name of the IN file parameter must be $RECEIVE.
   - FUP ignores the OUT file parameter, AUTOSTOP parameter, and any ASSIGN or PARAM messages.

FUP uses the default volume and subvolume in the startup message unless ZSCRATCH is specified only for the LOAD or LOADALTFILE commands. They are not used for the other commands.

The text portion of the startup message must be filled with zeros.

Figure 2-1. Communicating With FUP



3. After you send the startup message, close FUP.
4. Open FUP for programmatic use.

   Use #ZSPI as the first qualifier (in the process file name) when you open FUP. The process file name must have the format:

   $*process-name*.#ZSPI

   You can include backup OPENs and CLOSEs in FUP, but the first qualifier of the process file name for a backup OPEN must also be #ZSPI.

   Additional restrictions apply when you open the FUP server:

   - You can open FUP only if you created the FUP process.
   - You can issue only one concurrent open.
   - The SYNC and NOWAIT depth must be zero.

   Check for any file-system errors after you open FUP. File-system errors associated with applications using SPI include:

Error 11	FUP rejects the OPEN attempt because the first qualifier of the process file name is not #ZSPI.
Error 12	Your management process tried to open FUP more than once.
Error 17	A problem occurred during the backup OPEN. This can happen if the backup OPEN does not have a matching primary OPEN or if the backup and primary OPEN parameters do not match.
Error 48	FUP rejects the OPEN because your user ID does not have the proper security to perform the OPEN. This can happen when the process trying to open FUP is not the creator of the FUP process.

   Note. For a description of each file-system error, see the *Guardian Procedure Errors and Messages Manual.*

5. Allocate the command and response buffer.

   The recommended minimum buffer length in bytes is ZFUP-VAL-BUFLEN. A buffer of this length is large enough to hold the command and response for all FUP programmatic commands.

   Note. Although this manual refers to the SSINIT, SSNULL, and SSGET procedures, Compaq also provides the SSPUTTKN, SSGETTKN, and SSMOVETKN procedures. These procedures have the same functions as SSPUT, SSGET, and SSMOVE, and a calling sequence that is simpler for TAL programming. Compaq also provides the #SSINIT, #SSNULL, #SSPUT, #SSPUTV, #SSGET, #SSGETV, and #SSMOVE built-in functions for TACL programming.

6. Initialize the command buffer.

   Call the SSINIT procedure to specify the FUP command and initialize the buffer (including the addition of header tokens).

7. Initialize and set the fields of structured tokens.

   Call the SSNULL procedure to initialize the fields of each extensible structured token to null values, then set the individual fields of each structured token.

8. Add the tokens to the buffer.

   Call the SSPUT procedure for each token that you want to put in the buffer. Specify the command buffer, the unique token code, and a token value for each token. SSPUT places the token values in the buffer.

9. Send the buffer.

   Send the command buffer to FUP with the procedure that is appropriate for the language you are using (such as WRITEREAD for TAL, READ WITH PROMPT for COBOL85, or an #APPENDV/#REPLYV loop for TACL). If you have opened FUP for NOWAIT I/O, FUP accepts only one NOWAIT I/O operation.

   After you send the command buffer to FUP, check for file-system errors. File-system errors associated with applications using SPI include:

   Error 2 The request is not a correctly formatted SPI buffer. This can happen if the first two bytes do not contain -28.

   Error 60 You did not open FUP before you sent the command buffer.

   After FUP receives the buffer, it interprets the command request, executes the command (if no errors exist in the command format), and returns a response in the buffer (including any execution errors).

10. Interpret the response buffer from FUP.

    After FUP returns the response buffer, call the SSGET procedure to retrieve the tokens (including any error tokens) from the buffer. You must call SSGET for each token in the buffer.

11. Take the appropriate action.

    Decide what to do after checking the FUP reply in the response buffer. If FUP returned an error, execute an error-handling routine.

12. If the context token (ZSPI-TKN-CONTEXT) is present in the response record buffer, rebuild the buffer with this token and return to Step 9.

13. Check for another command request.

    If you need to send another command request to FUP, see Step 6.

14. Close FUP. (You can close FUP when you do not have any additional command requests. The FUP process stops when you close it.)

Starting and Opening FUP (TAL Example)

Figure 2-2 shows an example of a TAL integer procedure that is used to start and open a FUP process for programmatic use.

Figure 2-2. TAL Procedure to Start and Open a FUP Process (page 1 of 2)

```
!---------------------------------------------------------
! Procedure START^AND^OPEN^PUP^PROCESS starts a new FUP
! process and opens the process for programming use.
! Returns:  0 if successful, or an error number.
!---------------------------------------------------------
INT PROC START^AND^OPEN^FUP^PROCESS ( file^num );
INT .file^num;                  ! File number for FUP server process
BEGIN
! Local definitions
!INT error;                     ! Error for return
INT .S^PTR;                     ! End-of-string pointer (word pointer)
STRUCT .start^msg (st^msg^def);  ! Startup message (uses dummy template)
STRUCT .start^msg (st^msg^def);  ! Startup message (uses dummy template)

! Set up variables for process creation call
!
STRING PROG_NAME[0:ZSYS^VAL^LEN^FILENAME-1];
STRING PROCESS^NAME[0:ZSYS^VAL^LEN^FILENAME-1];
INT .EXT ERROR_DETAIL, OUTPUT_LIST_LEN;
STRUCT OUTPUT_LIST(ZSYS^DDL^SMSG^PROCCREATE^DEF);
STRUCT PARAM_LIST(PROCESS_LAUNCH_PARMS_);

! Initialize the structure
!
PARAM_LIST ':=' P_L_DEFAULT_PARMS_;

! Initialize the program file name
!
PROG_NAME ':=' "$SYSTEM.SYSnn.FUP" -> @S^PTR;
@PARAM_LIST.PROGRAM_NAME := $XADR ( PROG_NAME );
PARAM_LIST.PROGRAM_NAME_LEN := $DBL ( @S^PTR '-' @PROG_NAME );

! Initialize the process name
!
!process^name ':=' "$MFUP.#ZSPI" -> @S^PTR;
@PARAM_LIST.PROCESS_NAME := $XADR ( process^name );
PARAM_LIST.PROCESS_NAME_LEN := $DBL ( @S^PTR '-' @process^name );

! Build the startup message
!
start^msg.msgid := -1;              ! Startup message ID
start^msg.in    ':=' "$RECEIVE                ", ! IN file
start^msg.out   ':=
                0 & start^msg.txt FOR $LEN ( start^msg.txt ) - 1 BYTES;
```

Figure 2-2. TAL Procedure to Start and Open a FUP Process (page 2 of 2)

```
!Create the FUP process and check for an error
!
error := PROCESS_LAUNCH_ ( PARAM_LIST, ERROR_DETAIL, OUTPUT_LIST:$LEN (
OUTPUT_LIST ), OUTPUT_LIST_LEN );
IF error THEN RETURN error;

!Open the process and check for an error
!
error := FILE_OPEN_ ( process^name:$LEN( process^name ),file^num );
IF error THEN RETURN error;

! Send the startup message to the process and check for an error
!
CALL WRITEX ( file^num, start^msg, $OFFSET(start^msg.txt) );
IF <> THEN RETURN io^error (file^num);  ! io^error is a dummy
                                        ! error-handling proc

! Close the process and check for an error
!
error := FILE_CLOSE_ ( file^num );
IF error THEN RETURN error;

! Reopen the process for programming use
!
error := FILE_OPEN_ ( process^name:$LEN( process^name ), file^num );
IF error THEN RETURN error;
RETURN 0;                               ! Return with no errors
END;                                    ! End of procedure
```

Sending a Buffer to FUP (TAL Example)

Figure 2-3 shows an example of a TAL integer procedure that sends and receives a buffer using the WRITEREADX procedure.

Figure 2-3. TAL Procedure to Send a Buffer to FUP

```
!-----------------------------------------------------
! Procedure SEND^TO^SPI^PROCESS sends and receives a buffer
! using the WRITEREAD procedure.  The FUP process must al-
! ready be started and opened.  The buffer must be at least
! ZFUP^VAL^BUFLEN bytes.  This procedure returns 0 if it is
! successful, or an error number if it is not successful.
!-----------------------------------------------------
INT PROC SEND^TO^SPI^PROCESS (file^num, buffer);
INT file^num;    ! FUP server process file number
! SPI message buffer
STRUCT .EXT buffer (ZFUP^DDL^MSG^BUFFER^DEF);
BEGIN
! Local definitions
INT .EXT buffer^header (ZSPI^DDL^HEADER^DEF) = buffer;
INT  usedlen,
     initial^buffer^position;

! Get the used length token value in usedlen
CALL SSGETTKN (buffer, ZSPI^TKN^USEDLEN, usedlen);
! Send the buffer to FUP and check for a file-system error
CALL WRITEREADX (file^num, buffer, usedlen,
                 ZFUP^VAL^BUFLEN);
IF < THEN RETURN io^error (file^num);
! Check that all of the buffer was read
CALL SSGETTKN (buffer, ZSPI^TKN^USEDLEN, usedlen);
IF usedlen > ZFUP^VAL^BUFLEN THEN RETURN an^error;
! Reset length, position, and last error
buffer^header.Z^BUFLEN  := ZFUP^VAL^BUFLEN;
initial^buffer^position := ZSPI^VAL^INITIAL^BUFFER;
CALL SSPUTTKN (buffer, ZSPI^TKN^INITIAL^POSITION,
               initial^buffer^position);
CALL SSPUTTKN (buffer, ZSPI^TKN^CLEARERR);
RETURN 0;  ! Return with no errors
END;       ! End of procedure
```

After FUP returns the buffer, the TAL integer procedure in this example checks for a file-system error and verifies that the entire buffer was read.

Note. For more information on creating checks, see Receiving and Decoding a Response Buffer on page 2-13.

Elements of SPI Messages for FUP

A command and response buffer contains special codes called tokens. Each token contains a specific piece of information such as the command number or object type. Examples of these tokens are ZFUP-CMD-CHECKSUM, which is the command number token for the CHECKSUM command, and ZFUP-OBJ-FILE, which is the object-type token.

Source Definition Files

When you write your FUP management application, each source module must include the SPI standard definitions and the FUP definitions. Depending on the language you use, include these files with your source code (the disk volume is selected at your site):

C	Use the #INCLUDE directive to include the ZSPIDEF.ZSPIC and ZSPIDEF.ZFUPC files.
COBOL85	Use the COPY statement to include ZSPIDEF.ZSPICOB and ZSPIDEF.ZFUPCOB. Use the COPY statement with the REPLACING option to include a section of a file.

TACL	Load the ZSPIDEF.ZSPITACL and ZSPIDEF.ZFUPTACL files. To avoid buffer overflows during loading, load each file:
	`PUSH X`
	`#LOAD / LOADED X / $`*`volume`*`.ZSPIDEF.ZFUPTACL`
	`POP X`

TAL	Use the ?SOURCE directive to include ZSPIDEF.ZSPITAL and ZSPIDEF.ZFUPTAL.

Note. Except for the examples in TAL, the symbolic names in this manual are in DDL (or COBOL85) format using hyphens (-) as separators. If you are writing a TAL or TACL application, substitute the circumflex (^) symbol for the hyphens. If you are writing a C application, substitute the underscore (_) symbol for the hyphens.

Naming Rules for Applications

Compaq uses symbolic names beginning with the letter Z for all definitions (including the fields of structures) in definition files. To avoid conflicts with Compaq names, do not begin any names you define in your application with an uppercase or lowercase Z.

Common Syntax Elements

These syntax elements are common to FUP commands. Not every element is applicable to each command.

Command Numbers

Each FUP command is assigned a unique command number. The command numbers are represented by symbolic names using the form *ZFUP-CMD-name.* (The *name*

parameter identifies the command.) For example, the symbolic name for the LOAD command is ZFUP-CMD-LOAD.

Object Types

Each FUP command (except GETVERSION) supports the object type ZFUP-OBJ-FILE. GETVERSION supports the ZFUP-OBJ-NULL object type. FUP object-name tokens designate a file name or a file set. Table 2-1 shows the FUP programmatic commands and object types.

Note. For more information about object names, see SPI Programming Considerations for FUP on page 2-12.

Table 2-1. FUP Commands and Object Types

Command Name (ZFUP-CMD-)	Object Type (ZFUP-OBJ-)
GETVERSION	NULL
CHECKSUM	FILE
DUPLICATE	FILE
LOAD	FILE
LOADALTFILE	FILE
RESTART	FILE
Command Name (ZFUP-CMD-)	Object Type (ZFUP-OBJ-)

Additional FUP Command and Response Buffer Tokens

This subsection describes additional tokens used in FUP command or response buffers—including simple token codes, token types, predefined value names, field types, token maps, and structured tokens.

Simple Token Codes

These codes are represented by symbolic names using the form Z*sss*-TKN-*name* (where *sss* is the subsystem abbreviation, and *name* is the token code). For example, a simple token code for FUP is ZFUP-TKN-FILE, which identifies a file name or a file set for a command.

Note. For more information about the FUP token codes, see Section 3, FUP Commands and Responses.

Table 2-2. FUP Token Codes

Token Code (ZFUP-TKN-)	Token Type
ALTFILE-NUM	ZSPI-TYP-INT
BLOCKLEN	ZSPI-TYP-INT2
DEST-FILE	ZSPI-TYP-FNAME
FILE	ZSPI-TYP-FNAME
PART-NO	ZSPI-TYP-INT
PE-NUM	ZSPI-TYP-INT
READ-COUNT	ZSPI-TYP-INT4
RECLEN	ZSPI-TYP-INT2
REC-COUNT	ZSPI-TYP-INT4
SOURCE-FILE	ZSPI-TYP-FNAME

Token Types

These are represented by symbolic names using the form Z*sss*-TYP-*name* (where *sss* is the subsystem abbreviation, and *name* is the token type). FUP does not define any private token types (such as token types with the name ZFUP-TYP-*name*). Table 2-3 on page 2-11 shows the standard SPI token types used by FUP.

Predefined Value Names

These are represented by symbolic names using the form Z*sss*-VAL-*name* (where *sss* is the subsystem abbreviation, and *name* is the predefined value). An example of a predefined value name is ZFUP-VAL-BUFLEN, which is the recommended buffer length.

Token Maps and Structured Tokens

A token map is a variable-length integer array that contains decoding information and a reference name for an extensible structured token. A token map contains a token code and a description of the token value—including the token fields, the null values for the fields, and the version number for the fields. An application uses a token map to pass information in an extensible structured token to FUP.

Note. For more information about the FUP token maps, see Section 3, FUP Commands and Responses.

Field Types

FUP uses SPI standard field types that are represented using the form ZSPI-DDL-*name* (where *name* specifies the field type). For example, the symbolic name for the message buffer is ZFUP-DDL-MSG-BUFFER.

Event Messages

Although FUP does not report event messages to the Event Management Service (EMS), event messages can be generated by the NonStop Kernel or the disk process when you use the programmatic interface to FUP.

Note. For more information, see the *Operator Messages Manual*.

Standard SPI Token Types

Table 2-3. Standard SPI Token Types Used by FUP

Token Type	Description
ZSPI-TYP-BOOLEAN	16-bit signed Boolean value
ZSPI-TYP-BYTE	8-bit unsigned integer
ZSPI-TYP-BYTESTRING	String of 8-bit unsigned integers
ZSPI-TYP-CHAR	8-bit ASCII character
ZSPI-TYP-CHAR-PAIR	Pair of 8-bit ASCII characters
ZSPI-TYP-CHAR24	Twenty-four 8-bit ASCII characters
ZSPI-TYP-CHAR50	Fifty 8-bit ASCII characters
ZSPI-TYP-DEVICE	8-byte internal device name
ZSPI-TYP-ENUM	16-bit signed enumerated value
ZSPI-TYP-ERROR	SPI error token
ZSPI-TYP-FLT2	64-bit floating-point number
ZSPI-TYP-FNAME	24-byte internal file name
ZSPI-TYP-INT	16-bit signed integer
ZSPI-TYP-INT2	32-bit signed integer
ZSPI-TYP-INT2-PAIR	Pair of 32-bit signed integers
ZSPI-TYP-INT4	64-bit fixed-point number
ZSPI-TYP-LIST	Token starting a list
ZSPI-TYP-POSITION	64-bit SPI position descriptor
ZSPI-TYP-SSCTL	Special SPI control operation
ZSPI-TYP-SSID	Subsystem ID
ZSPI-TYP-STRING	Variable-length ASCII string
ZSPI-TYP-SUBVOL	16-byte internal subvolume name

SPI Programming Considerations for FUP

The *SPI Programming Manual* provides general programming considerations for management applications that use SPI command and response buffers to communicate with subsystems such as FUP. This subsection provides considerations specific to FUP.

Building the Command Buffer

Your application must allocate a command and response buffer to communicate with FUP. This buffer must be large enough to hold each FUP command and response. (Multiple FUP commands in a single buffer are not allowed.) The recommended minimum buffer size in bytes is ZFUP-VAL-BUFLEN.

Your application uses SPI procedures to initialize the command buffer and add tokens to it. Use the SSINIT procedure to initialize the buffer and to specify the FUP command. SSINIT also adds the header tokens to the buffer.

When the buffer is initialized with SSINIT, any previous contents of the buffer are overwritten. Therefore, to save the contents of the buffer before calling SSINIT, save a copy of the buffer in another location.

You use the SSNULL procedure to initialize the fields of extensible structured tokens with null values. Always call SSNULL, even if you explicitly set all currently defined fields of the structure. This ensures that your application will continue to run correctly if future versions of FUP add fields to these structured tokens.

Specifying Object Names

Each FUP programmatic command (except GETVERSION) requires an object type of ZFUP-OBJ-FILE. The object is either sent to FUP as a single file name, or (for the CHECKSUM and DUPLICATE commands) as a template that represents a file set. If the restartable option is specified for the DUPLICATE command, a single file name must be specified as the object.

Specifying a File Name

A file name sent to FUP in a command buffer must be in the *Guardian* internal file-name format and must be fully qualified. If the file is not on the same system as FUP, the name must be in the Guardian internal file-name format. If a file name does not include a system identifier, FUP assumes that the file is on the same system where FUP is running.

All file names returned by FUP in the response buffer are also in the Guardian internal file-name format.

Note. For more information about this format, see the *Guardian Programmer's Guide*.

Specifying a File Set

The CHECKSUM command and the DUPLICATE command (without the restartable option specified) support a template using the asterisk (*) to specify a file set within a volume or subvolume. A file set can be a single file, all files in a subvolume, or all files in a volume. You cannot use the asterisk to specify the volume name or the subvolume name when you specify a file name.

Table 2-4 shows examples of file-set templates. The byte position values in this table indicate the positions for the NonStop Kernel internal file-name format.

Table 2-4. FUP Files and File Sets

Byte Position			
0	8	16	Files Described
$VOLbbbbSUBVOLbbFILEbbbb			A single file
$VOLbbbbSUBVOLbb*bbbbbbb			All files in the subvolume
$VOLbbbb*bbbbbbb*bbbbbbb			All files in the volume
$VOLbbbb*bbbbbbbFILEbbbb			None; name invalid
*bbbbbbb*bbbbbbb*bbbbbbb			None; name invalid

Discontinuing a Command in Progress

FUP checks for CANCEL requests between the processing of each file for a multiple-file command. To cancel the DUPLICATE and CHECKSUM commands, use the CANCEL file-system procedure.

Note. For more information, see the *Guardian Procedure Calls Reference Manual*.

Receiving and Decoding a Response Buffer

When your application receives the response buffer, first check for these errors and execute an error routine (if an error occurs):

- Errors that were reported by the method used to send and receive the buffer, such as a file-system error
- Return token ZSPI-TKN-RETCODE for a value indicating that FUP found an error in the command
- Used-length token ZSPI-TKN-USEDLEN from the response buffer, to ensure that the buffer is not larger than the buffer allocated by your application

You should also check the values of these tokens in the SPI message header to ensure that the response matches the original command:

ZSPI-TKN-COMMAND	FUP command from the request.
ZSPI-TKN-OBJECT-TYPE	Object type from the request.
ZSPI-TKN-SSID	Subsystem ID of the FUP server that performed the command.

If no errors are found, you can extract the remaining tokens from the buffer and continue processing.

Processing a Single Response Record

The default FUP command and response is a single FUP command that generates one response record per reply buffer. The response record describes the action of the command on one file.

Figure 2-4. FUP Single Response Record

```
ZSPI-TKN-RETCODE
ZFUP-TKN-FILE
token-1
token-2
   .
   .
   .
token-n
error-list (if an error occurs)
```

Table 2-5. Tokens in a Single FUP Response Record

ZSPI-TKN-RETCODE token	This token reports any errors or warnings. A value of zero indicates the command was successful. A value other than zero indicates an error or warning occurred.
ZFUP-TKN-FILE token	This token identifies the object of the command (such as the file or file set).
FUP response tokens	These tokens (token-1, token-2, through token-*n*) are specific to the FUP command.
Error lists	If any errors or warnings occurred, the error or warning information is enclosed in one or more error lists.

Continuing a FUP Command

The CHECKSUM and DUPLICATE commands (without the restartable option specified) accept a file-set template for the source file name. Although this allows each command to process more than one file, each reply message applies only to a single file (by default). This causes each reply to contain the response for the file that FUP processed most recently and the ZSPI-TKN-CONTEXT context token. This token indicates that FUP has not finished processing all the files in the file set.

To checksum or duplicate the next file in the file set, return the original command buffer and ZSPI-TKN-CONTEXT to FUP. FUP uses ZSPI-TKN-CONTEXT to determine which file to process next. After FUP processes the last file in the file set, it returns a reply without ZSPI-TKN-CONTEXT.

Your application is not required to continue a command when a response record buffer contains the ZSPI-TKN-CONTEXT token. You can issue other commands before returning the command with ZSPI-TKN-CONTEXT to FUP, or you can abandon the continuation of the command altogether.

Processing an Empty Response

If FUP receives a command with the ZSPI-TKN-CONTEXT token and all of the files in the file set have been processed, FUP returns a response record with ZSPI-TKN-RETCODE set to ZFUP-ERR-EMPTY-RESP (specifying an empty response). FUP returns an empty response if the last file in a file set is deleted between the time FUP returned the previous response record and the time the application sent another command request with the ZSPI-TKN-CONTEXT token.

Processing Multiple Response Records in a Reply Message

Your application can request multiple response records per reply by setting the SPI token ZSPI-TKN-MAXRESP. This token specifies the maximum number of response records your application accepts in a single reply buffer. Multiple response records in a single reply message can reduce message traffic and improve system performance. The values for ZSPI-TKN-MAXRESP are:

0	FUP returns one response record per buffer (not enclosed in a data list). The default is zero.
-1	FUP returns as many response records per buffer as the buffer can hold (with each response record enclosed in a data list).
n	FUP returns *n* response records per buffer (with each response record enclosed in a data list). *n* must be a positive number.

If you specify a value other than zero for ZSPI-TKN-MAXRESP, FUP returns each response record in the buffer in a data list. See Figure 2-5.

Figure 2-5. Multiple FUP Response Records

```
datalist-1
datalist-2
   .
   .
   .
datalist-n
ZSPI-TKN-CONTEXT (if this is not the last reply message)
```

Each data list begins with ZSPI-TKN-DATALIST and ends with ZSPI-TKN-ENDLIST. Within each data list, the format is the same as the format for a single response record.

Each data list contains a ZSPI-TKN-RETCODE token to indicate the error status of the response for a specific file.

Figure 2-6 on page 2-16 shows a single response record in a data list. The error list is present only if an error occurred, as indicated by ZSPI-TKN-RETCODE.

Figure 2-6. Single FUP Record in a Data List

```
ZSPI-TKN-DATALIST
   ZSPI-TKN-RETCODE
   ZFUP-TKN-FILE
   token-1
   token-2
      .
      .
      .
   token-n
   error-list (if an error occurred)
ZSPI-TKN-ENDLIST
```

Warnings that apply to the FUP command (and not to the file) are returned in the first response record. If the command can be continued, the reply buffer contains the context token ZSPI-TKN-CONTEXT, which is not part of a data list. Similar to the single response record, ZSPI-TKN-CONTEXT indicates whether FUP has returned the last response record. Each request has a value for ZSPI-TKN-MAXRESP (either the specified value or zero if a value is not specified). FUP does not save this value for subsequent requests.

Controlling Types of Response Records

The SPI token ZSPI-TKN-RESPONSE-TYPE controls the types of response records that FUP returns in the buffer. The values for ZSPI-TKN-RESPONSE-TYPE are:

`ZSPI-VAL-ERR-AND-WARN`

FUP returns a response record only for a file that caused an error or a warning. The response buffer contains at least one error list, regardless of the value of ZSPI-TKN-RETCODE.

`ZSPI-VAL-ERR-WARN-AND-NORM`

FUP returns a response record for each file processed. The default is ZSPI-VAL-ERR-WARN-AND-NORM.

If the value of ZSPI-TKN-RESPONSE-TYPE is ZSPI-VAL-ERR-AND-WARN, and the command does not encounter any errors or warnings, FUP returns an empty response (with ZSPI-TKN-RETCODE set to ZFUP-ERR-EMPTY-RESP).

Note. For more information about an empty response, see Tokens in a Single FUP Response Record on page 2-14

If the value of ZSPI-TKN-RESPONSE-TYPE is ZSPI-VAL-ERR-AND-WARN, and a warning occurs for the command (not caused by the execution of the command on a file), FUP holds the warning until it generates a response caused by an error or warning for one of the files. If no warnings or errors are generated for any of the files, FUP returns the command warning in an empty response.

Handling FUP Errors

If ZSPI-TKN-RETCODE contains a value other than zero, an error or warning occurred, and the response buffer contains one or more error lists.

Figure 2-7. Contents of an Error List

```
ZSPI-TKN-ERRLIST
   ZSPI-TKN-ERROR
   token-1
   token-2
      .
      .
      .
   token-n
   nested error list, if another subsystem error occurs
ZSPI-TKN-ENDLIST
```

An error list begins with ZSPI-TKN-ERRLIST and ends with ZSPI-TKN-ENDLIST. The error token ZSPI-TKN-ERROR, which is always included in the error list, contains the FUP subsystem ID and the error number. Other tokens in the error list describe different aspects of the error such as the FUP command number and the file FUP was processing when the error occurred.

FUP returns a second (and sometimes third) nested error list when the error originates from another subsystem or software component (such as the FASTSORT utility or the Guardian file system).

Note. For a description of all of the FUP error lists, see Appendix B, FUP Error Messages.

The empty response occurs when ZSPI-TKN-RETCODE contains a value other than zero and the response buffer does not contain an error list. FUP returns an empty response when the command buffer contains the context token ZSPI-TKN-CONTEXT, but no more files remain to process.

Note. For more information about an empty response, see Tokens in a Single FUP Response Record on page 2-14.

If ZSPI-TKN-RETCODE is zero, the response buffer can still contain an error list that describes a warning condition. The warning condition did not prevent FUP from performing the requested command. To determine the warning, you must check the value of the ZSPI-TKN-ERROR token in the error list.

Types of FUP Errors

When your application sends a command buffer to FUP, these types of errors can be returned:

- Syntax errors in the FUP command format
- Command failure errors encountered by FUP
- Command failure errors encountered by a subsystem or software component other than FUP

Syntax Errors in the Command Format

FUP evaluates the command format in the buffer to determine if any syntax errors exist. Examples of these errors are an invalid token value or an invalid command. A FUP error list for a syntax error contains the ZSPI-TKN-ERROR token, which identifies the error and the FUP SSID, and the ZSPI-TKN-PARM-ERR token, which contains information about the syntax error. Because the command failed before FUP processed the file, a file name is not included in the error list.

An example of an invalid token value error occurs if a field within a structured token is out of range. ZSPI-TKN-RETCODE has a value of ZFUP-ERR-INV-VALUE (7) to indicate this error. See Figure 2-8.

Figure 2-8. Error List for a Syntax Error

```
ZSPI-TKN-ERRLIST
  ZSPI-TKN-ERROR
    Z-SSID           ! FUP subsystem ID
    Z-ERROR          ! ZFUP-ERR-INV-VALUE (7)
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE      ! Token code that caused the error
    Z-INDEX          ! Occurrence number of the token
    Z-OFFSET         ! Byte offset of the field that
                     ! caused the error
ZSPI-TKN-ENDLIST
```

FUP Errors

These errors are found when FUP attempts to execute the command. Examples of these errors are an empty source file or a command that is not allowed for the file type. The error list contains the token ZSPI-TKN-ERROR and a token map ZFUP-MAP-CMD-ERROR, which identifies the command that failed, the object type, and the file that FUP was processing when the error occurred.

An example of this occurs if the empty source file error reports that a source file is empty (contains no records) for an attempted LOAD command, with the ZEMPTYOK option not specified. ZSPI-TKN-RETCODE has a value of ZFUP-ERR-EMPTY-SOURCE (35) to indicate this error. For a sample of this type of error list, see Figure 2-9 on page 2-19.

Figure 2-9. Error List for a FUP Error

```
ZSPI-TKN-ERRLIST
  ZSPI-TKN-ERROR
    Z-SSID           ! FUP subsystem ID
    Z-ERROR          ! ZFUP-ERR-EMPTY-SOURCE (35)
  ZFUP-MAP-CMD-ERROR
    ZCOMMAND         ! LOAD command
    ZOBJECT          ! Object type (file)
    ZNAME            ! Name of the empty source file
ZSPI-TKN-ENDLIST
```

Other Subsystem and Software Component Errors

When FUP executes a command, these subsystems and software components are sometimes called to perform various tasks:

- EDITREAD and EDITREADINIT file-system procedures
- FASTSORT utility
- NonStop Kernel
- Guardian file system

Errors can originate from these subsystems or software components. When this type of error occurs, ZSPI-TKN-RETCODE indicates that the attempted command failed with an error from a source other than FUP. The FUP error list contains the token ZSPI-TKN-ERROR, the token map ZFUP-MAP-CMD-ERROR, and another nested error list describing the actual error. The nested error list indicates where the error occurred.

If a LOAD command failed because of a file-system error on a WRITE procedure call, ZSPI-TKN-RETCODE has the value ZFUP-ERR-FILESYS (17) to indicate this error. Figure 2-10 on page 2-20 shows an example of this nested error list.

Figure 2-10. File-System Nested Error List

```
ZSPI-TKN-ERRLIST  ! Start of FUP error list
  ZSPI-TKN-ERROR
    Z-SSID        ! FUP subsystem ID
    Z-ERROR       ! ZFUP-ERR-FILESYS (17)
  ZFUP-MAP-CMD-ERROR
    ZCOMMAND      ! LOAD command
    ZOBJECT       ! Object type (file)
    ZNAME         ! Name of the file that FUP was
                  ! processing when the error occurred

  ZSPI-TKN-ERRLIST      ! Start of file-system error list
    ZSPI-TKN-ERROR
      Z-SSID            ! File-system ID
      Z-ERROR           ! File-system error number
      ZSPI-TKN-PROC-ERR ! WRITE procedure call
  ZSPI-TKN-ENDLIST      ! End of file-system error list

ZSPI-TKN-ENDLIST  ! End of FUP error list
```

Errors from the FASTSORT utility are more complicated than file-system errors. The FASTSORT utility can fail because of an error from another subsystem or software component, causing nested error lists. The FUP error list contains the FASTSORT error list, and the FASTSORT error list contains another error list that describes where the error actually occurred.

If a LOAD command failed because the FASTSORT utility could not allocate an extended segment after a NonStop Kernel procedure call, then ZSPI-TKN-RETCODE has the value ZFUP-ERR-SORT (16) to indicate that the command failed with a FASTSORT error. An example of this nested error list is displayed in Figure 2-11 on page 2-21.

Figure 2-11. FASTSORT/NonStop Kernel Nested Error List

```
ZSPI-TKN-ERRLIST   ! Start of FUP error list
  ZSPI-TKN-ERROR
    Z-SSID         ! FUP subsystem ID
    Z-ERROR        ! ZFUP-ERR-SORT (16)
  ZFUP-MAP-CMD-ERROR
    ZCOMMAND       ! LOAD command
    ZOBJECT        ! Object type (file)
    ZNAME          ! Name of the file that FUP was
                   ! unable to load

  ZSPI-TKN-ERRLIST ! Start of FASTSORT error list
    ZSPI-TKN-ERROR
      Z-SSID       ! FASTSORT utility ID
      Z-ERROR      ! FASTSORT error (64)

    ZSPI-TKN-ERRLIST     ! Start Compaq NonStop error list
      ZSPI-TKN-ERROR
        Z-SSID           ! Compaq NonStop OS SSID
        Z-ERROR          ! File-system error 43
        ZSPI-TKN-PROC-ERR ! Segment allocation procedure
    ZSPI-TKN-ENDLIST    ! End of Compaq NonStop error list

  ZSPI-TKN-ENDLIST ! End of FASTSORT error list

ZSPI-TKN-ENDLIST   ! End of FUP error list
```

To extract tokens from the nested error lists, use SSGET calls or loop over the buffer and get all of the sequential tokens using ZSPI^TKN^NEXTTOKEN, then use a CASE statement to process the expected tokens. Figure 2-12 on page 2-22 shows how to navigate a nested error list.

Note. Figure 2-12 does not show error checking.

Figure 2-12. Extracting Tokens From a Nested Error List (page 1 of 2)

```
! Get the FASTSORT error
CALL SSGETTKN (buffer,
               ZSPI^TKN^ERROR,
               sort^error, 1);         .
          .
          .
! Extracting tokens from a nested error list

! Variables to extract a token map from the buffer
INT .cmd^error^resp^map [0:(ZFUP^MAP^CMD^ERROR^WLN-1)] :=
        ZFUP^MAP^CMD^ERROR;

STRUCT .error^info (ZFUP^DDL^CMD^ERROR^DEF);
          .
          .
          .
! Get the return token from the response buffer
CALL SSGETTKN (buffer,
               ZSPI^TKN^RETCODE,
               return^code, 1);
          .
          .
          .
! ZSPI^TKN^RETCODE indicates an error
          .
          .
          .
! Enter the FUP error list
CALL SSGETTKN (buffer,
               ZSPI^TKN^ERRLIST,, 1);

! Get the FUP subsystem ID and error
CALL SSGETTKN (buffer,
               ZSPI^TKN^ERROR,
               fup^error, 1);

! Get the command error information
CALL SSGET (buffer,
            cmd^error^resp^map,
            error^info, 1);
          .
          .
          .
! FUP error list indicates a FASTSORT error

! Enter the FASTSORT error list
CALL SSGETTKN (buffer,
               ZSPI^TKN^ERRLIST,, 1);
```

Figure 2-12. Extracting Tokens From a Nested Error List (page 2 of 2)

```
! Get the FASTSORT error
CALL SSGETTKN (buffer,
               ZSPI^TKN^ERROR,
               sort^error, 1);         .
          .
          .
! FASTSORT error list indicates a Compaq NonStop error

! Enter the Compaq NonStop error list
CALL SSGETTKN (buffer,
            ZSPI^TKN^NEXTTOKEN,
            !token!,            -- Should be ZSPI^TKN^ERRLIST
            !index!,            -- Getting the next one
            !count!,            -- Only expect one token
            guardian^ssid);     -- Get Compaq NonStop SSID

! Get the Compaq NonStop error
CALL SSGETTKN (buffer,
            ZSPI^TKN^ERROR,
            guardian^error,
            1,
            !count!,
            guardian^ssid);     -- Get Compaq NonStop SSID

! Get the Compaq NonStop procedure
CALL SSGETTKN (buffer,
            ZSPI^TKN^PROC^ERR,
            guardian^proc,
            1,
            !count!,
            guardian^ssid);     -- Get Compaq NonStop SSID
          .
          .
          .
! Exit the Compaq NonStop error list
CALL SSGETTKN (buffer,
            ZSPI^TKN^ENDLIST,
            !token!,
            !index!,
            !count!,
            guardian^ssid);      -- Get Compaq NonStop SSID

! Exit the FASTSORT error list
CALL SSGETTKN (buffer, ZSPI^TKN^ENDLIST);

! Exit the FUP error list
CALL SSGETTKN (buffer, ZSPI^TKN^ENDLIST);

! Call a procedure to report the error
```

Continuing After an Error

An error can occur on only one of the files in a file set for the commands that process more than one file (CHECKSUM and DUPLICATE without the restartable option). If an error does occur on a file, the SPI token ZSPI-TKN-ALLOW-TYPE determines if FUP should continue processing the remaining files in the file set. The values for ZSPI-TKN-ALLOW-TYPE are:

ZSPI-VAL-NORM-ONLY	FUP processes the next file only if the command was successful for the previous file. (FUP did not return an error list for the file.) This is the default.
ZSPI-VAL-WARN-AND-NORM	FUP processes the next file if the command completed for the previous file. A warning might have occurred, but the ZSPI-TKN-RETCODE value is zero.
ZSPI-VAL-ERR-WARN-AND-NORM	FUP processes the next file despite any problems encountered for the previous file.

The value of ZSPI-TKN-ALLOW-TYPE is significant only if ZSPI-TKN-MAXRESP is -1 (or is greater than 1). Otherwise FUP ignores the token.

Common Definitions

The FUP programmatic commands use SPI standard definitions and FUP definitions. This subsection provides a general description of these definitions. For specific information about each definition, see Section 3, FUP Commands and Responses or Appendix B, FUP Error Messages.

Note. All definitions are shown in DDL (or COBOL85) format using hyphens (-) as separators. If you are programming in TAL or TACL, substitute the circumflex (^) symbol for the hyphens. If you are programming in C, substitute the underscore (_) symbol for the hyphens.

SPI Standard Definitions

SPI standard definitions begin with ZSPI- and are found in the ZSPITAL, ZSPICOB, ZSPITACL, ZSPIPAS, and ZSPIC source definition files.

Note. For more information about these files, see Source Definition Files on page 2-8.

Table 2-6. SPI Standard Definitions Used by FUP (page 1 of 2)

Header Tokens		
ZSPI-TKN-CHECKSUM	ZSPI-TKN-LASTPOSITION	ZSPI-TKN-SERVER-VERSION
ZSPI-TKN-COMMAND	ZSPI-TKN-MAX-FIELD-VERSION	ZSPI-TKN-SSID
ZSPI-TKN-HDRTYPE	ZSPI-TKN-MAXRESP	ZSPI-TKN-USEDLEN
ZSPI-TKN-LASTERR	ZSPI-TKN-OBJECT-TYPE	

Table 2-6. SPI Standard Definitions Used by FUP (page 2 of 2)

ZSPI-TKN-LASTERRCODE	ZSPI-TKN-POSITION	
Special Tokens		
ZSPI-TKN-ADDR	ZSPI-TKN-INITIAL-POSITION	ZSPI-TKN-OFFSET
ZSPI-TKN-CLEARERR	ZSPI-TKN-LEN	ZSPI-TKN-RESET-BUFFER
ZSPI-TKN-COUNT	ZSPI-TKN-NEXTCODE	
ZSPI-TKN-DEFAULT-SSID	ZSPI-TKN-NEXTTOKEN	
Other Simple Tokens		
ZSPI-TKN-ALLOW-TYPE	ZSPI-TKN-ENDLIST	ZSPI-TKN-RESPONSE-TYPE
ZSPI-TKN-COMMENT	ZSPI-TKN-ERRLIST	ZSPI-TKN-RETCODE
ZSPI-TKN-CONTEXT	ZSPI-TKN-ERROR	ZSPI-TKN-SERVER-BANNER
ZSPI-TKN-DATALIST	ZSPI-TKN-PARM-ERR	
Value Names		
ZSPI-SSN-ZFUP	ZSPI-VAL-FALSE	ZSPI-VAL-TANDEM
ZSPI-VAL-TRUE		
Token Types		
ZSPI-TYP-BOOLEAN	ZSPI-TYP-ENUM	ZSPI-TYP-SSCTL
ZSPI-TYP-BYTE-PAIR	ZSPI-TYP-ERROR	ZSPI-TYP-SSID
ZSPI-TYP-BYTESTRING	ZSPI-TYP-FNAME32	ZSPI-TYP-STRING
ZSPI-TYP-CHAR8	ZSPI-TYP-INT	ZSPI-TYP-TIMESTAMP
ZSPI-TYP-CHAR50	ZSPI-TYP-LIST	ZSPI-TYP-UINT
ZSPI-TYP-CRTPID	ZSPI-TYP-MARK	
ZSPI-TYP-DEVICE	ZSPI-TYP-PARM-ERR	
Structures		
ZSPI-DDL-BOOLEAN	ZSPI-DDL-ENUM	ZSPI-DDL-INT2
ZSPI-DDL-BYTE	ZSPI-DDL-FNAME	ZSPI-DDL-TIMESTAMP
ZSPI-DDL-CHAR8	ZSPI-DDL-FNAME32	ZSPI-DDL-UINT
ZSPI-DDL-CRTPID	ZSPI-DDL-INT	
ZSPI-DDL-DEVICE	ZSPI-DDL-INT-PAIR	

These SPI standard definitions are specific to FUP:

Note. For each SPI standard definition, see the *SPI Programming Manual*.

ZSPI-SSN-ZFUP

is the subsystem number assigned to FUP.

ZSPI-TKN-COMMAND

contains the command number for these FUP programmatic commands: CHECKSUM, DUPLICATE, GETVERSION, LOAD, LOADALTFILE, or RESTART.

ZSPI-TKN-ERROR

is the error token that is present in an error list. This token contains the FUP subsystem ID and error number.

For more information about all FUP numbers and their associated error lists, see Appendix B, FUP Error Messages.

ZSPI-TKN-PARM-ERR

is a parameter error token that is present in some error lists. This token identifies a token code (Z-TOKENCODE), the occurrence number of the token (Z-INDEX), and the byte offset of a specific field in the token (Z-OFFSET).

ZSPI-TKN-OBJECT-TYPE

contains the object-type number for the FUP object. The object-type for FUP commands is ZFUP-OBJ-FILE (except for the GETVERSION command which is ZFUP-OBJ-NULL).

ZSPI-TKN-RETCODE

is the return token. For a description of each FUP error number and its corresponding error list, see Appendix B, FUP Error Messages. For ZSPI-TKN-RETCODE values common to all FUP commands, see Table 2-7 on page 2-28.

For a list of more ZSPI-TKN-RETCODE values with their corresponding command, see Section 3, FUP Commands and Responses.

ZSPI-TKN-SERVER-VERSION

contains the server version of the FUP subsystem.

ZSPI-TKN-SSID

contains ZFUP-VAL-SSID, the subsystem ID of the FUP subsystem. ZFUP-VAL-SSID has this structure:

```
def ZFUP-VAL-SSID tacl ssid.
   02 Z-FILLER            type character 8
                          value is ZSPI-VAL-TANDEM.
   02 Z-OWNER             redefines Z-FILLER
                          type ZSPI-DDL-CHAR8.
   02 Z-NUMBER            type ZSPI-DDL-INT
                          value is ZSPI-SSN-ZFUP.
   02 Z-VERSION           type ZSPI-DDL-UINT
                          value is ZFUP-VAL-VERSION.
end.
```

FUP Definitions

FUP definitions begin with ZFUP- and are found in the source definition files ZFUPTAL, ZFUPCOB, ZFUPTACL, and ZFUPC. For more information about FUP definitions, see Section 3, FUP Commands and Responses.

FUP Message Buffer Declaration

ZFUP-DDL-MSG-BUFFER

is the SPI buffer you use for the FUP commands. For example:

```
def ZFUP-DDL-MSG-BUFFER.
   02 Z-MSGCODE          type ZSPI-DDL-INT.
   02 Z-BUFLEN           type ZSPI-DDL-UINT.
   02 Z-OCCURS           type ZSPI-DDL-UINT.
   02 Z-FILLER           type ZSPI-DDL-BYTE
                         occurs 0 TO ZFUP-VAL-BUFLEN times
                         depending on Z-OCCURS.
end.
```

Predefined Token and Field Values

ZFUP-VAL-BUFLEN

is the recommended buffer length in bytes for all FUP command buffers.

ZFUP-VAL-SSID

is the FUP subsystem ID.

ZFUP-VAL-VERSION

is the version number of the FUP subsystem.

For more information about predefined token and field values, see Section 3, FUP Commands and Responses.

Simple Tokens and Structured Tokens

For descriptions of these tokens for each FUP command, see Section 3, FUP Commands and Responses.

Tokens in Error Lists

These tokens are used in FUP error lists. For more information about FUP error lists, see Handling FUP Errors on page 2-17.

ZFUP-MAP-CMD-ERROR

contains these fields:

ZNAME

is the name of the file that FUP was processing (or attempting to process) when the error occurred.

ZCOMMAND

is the FUP command that was executing when the error occurred.

ZOBJECT

is the object type for the FUP command.

Note. For more information about all FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages.

Table 2-7. Errors Returned by All FUP Commands

Error Number	Symbolic Name (ZFUP-ERR-)	Description
0	OK	The command finished successfully.
1	INV-COMMAND	FUP found an invalid command.
2	INV-OBJECT	FUP found an invalid object type.
3	INVALID-TOKEN	FUP found an invalid token.
4	MISS-TOKEN	FUP detected a missing token.
5	MISS-FIELD	FUP detected a missing field.
6	EXTRA-TOKEN	FUP found an extra token.
7	INV-VALUE	FUP found an invalid token or field.
8	INV-CONTEXT	FUP found an invalid context token.
9	INV-TEMPLATE	FUP found an invalid template.
10	LONG-COMMAND	A command was too long.
11	WRONG-SSID	The application specified an invalid FUP subsystem ID.
12	WRONG-SERVER	The application specified an invalid FUP server version.
13	EMPTY-RESP	FUP returned an empty response.
14	NO-MEM	Insufficient memory was available.
15	EDITREAD	An EDITREAD error occurred.
16	SORT	A FASTSORT error occurred.
17	FILESYS	A file-system error occurred.
18	GUARD	A NonStop Kernel error occurred.
19	SPI	An SPI subsystem error occurred.
20	PE	A programming error occurred.

3

FUP Commands and Responses

This section describes the FUP programmatic commands and responses for:

Each description contains:

- A header showing the command name
- A summary of the function of the command
- A box that lists these elements for each command:
  - The symbolic name for the command number
  - The symbolic name of the object type
  - A list of tokens that can be used in the command buffer
  - A list of tokens that FUP can return in the response buffer
- A description of tokens listed in the box
- Considerations for using the command
- Examples of the command

While reading the descriptions, consider:

- Although the list of the tokens in the box is not necessarily in the order that the tokens actually appear in a command or response buffer, the token ZSPI-TKN-ENDLIST always appears at the end of a list started by ZSPI-TKN-DATALIST or ZSPI-TKN-ERRLIST.
- The notation used in the box for simple tokens is a shortened version of the DDL TOKEN-CODE statement. To define structured tokens, use the DDL DEF statement.

CHECKSUM Command

The CHECKSUM command recomputes the checksum value for blocks of data in disk files.

```
Command

ZFUP-CMD-CHECKSUM

Object Type

ZFUP-OBJ-FILE
```

```
Tokens in the Command Buffer

ZFUP-TKN-FILE            token-type ZSPI-TYP-FNAME.

ZFUP-MAP-PAR-CHECKSUM

def ZFUP-DDL-PAR-CHECKSUM.
  02  ZPART-ONLY         type ZSPI-DDL-BOOLEAN.
end.

ZSPI-TKN-MAXRESP         token-type ZSPI-TYP-INT.
ZSPI-TKN-CONTEXT         token-type ZSPI-TYP-BYTESTRING.
ZSPI-TKN-RESPONSE-TYPE   token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT         token-type ZSPI-TYP-STRING.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST        token-type ZSPI-TYP-LIST.
  ZFUP-TKN-FILE          token-type ZSPI-TYP-FNAME.
  ZSPI-TKN-RETCODE       token-type ZSPI-TYP-RETCODE.
  ZSPI-TKN-ERRLIST       token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST     token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST       token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-CONTEXT         token-type ZSPI-TYP-BYTESTRING.
```

Tokens in the Command Buffer

ZFUP-TKN-FILE

is a required token that specifies a file (or file set) whose checksum values must be recomputed. The file name must be in the Guardian internal file-name format, and only one of these tokens is allowed for each command.

ZFUP-MAP-PAR-CHECKSUM

is an optional structured token that specifies whether primary or secondary partitions of a partitioned file are to have the checksum value recomputed. Only one of these tokens is allowed per command.

ZPART-ONLY

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies to have the checksum value recomputed for only the primary or secondary partitions of a partitioned file (defined by the file-set template). This field does not affect nonpartitioned files.

ZSPI-VAL-FALSE

specifies to have the checksum value recomputed for any primary partitions of a partitioned file, and all secondary partitions of the files (the entire file) that are defined by the fileset template. The default is ZSPI-VAL-FALSE.

ZSPI-TKN-MAXRESP

is the standard SPI token that indicates the maximum number of response records that FUP returns in a response buffer. The values for ZSPI-TKN-MAXRESP are:

0	FUP returns one response record per buffer (not enclosed in a data list). The default is zero.
1	FUP returns as many response records per buffer as the buffer can hold (with each response record enclosed in a data list).
n	FUP returns *n* response records per buffer (with each response record enclosed in a data list). *n* must be a positive number.

ZSPI-TKN-CONTEXT

is the standard SPI token that indicates whether FUP has more reply messages to return:

- If ZSPI-TKN-CONTEXT is present in the response buffer, FUP has more reply messages to return. Return the token in the command buffer to FUP. You can ignore the actual value of ZSPI-TKN-CONTEXT because this value is important only to FUP.
- If ZSPI-TKN-CONTEXT is not present in the response buffer, FUP has returned all of the reply messages.

ZSPI-TKN-RESPONSE-TYPE

is the standard SPI token that indicates the type of response records that FUP returns. The values for ZSPI-TKN-RESPONSE-TYPE are:

ZSPI-VAL-ERR-AND-WARN

FUP returns a response record only for a file that caused an error or a warning. The response buffer contains at least one error list (despite the value of ZSPI-TKN-RETCODE).

ZSPI-VAL-ERR-WARN-AND-NORM

FUP returns a response record for each file processed. This is the default.

ZSPI-TKN-ALLOW-TYPE

is the standard SPI token that determines if FUP continues processing the remaining files in a file set (despite the occurrence of an error on a current file in the file set).

ZSPI-TKN-ALLOW-TYPE is significant only if ZSPI-TKN-MAXRESP is -1 (or greater than 1). Otherwise, FUP ignores the token. The values for ZSPI-TKN-ALLOW-TYPE are:

ZSPI-VAL-NORM-ONLY

FUP processes the next file only if the command was successful for the current file (that is, FUP did not return an error list for the file). This is the default.

ZSPI-VAL-WARN-AND-NORM

FUP processes the next file if the command completed for the current file. A warning might have occurred for the file, but the ZSPI-TKN-RETCODE value is zero.

ZSPI-VAL-ERR-WARN-AND-NORM

FUP processes the next file regardless of any problems encountered for the current file.

ZSPI-TKN-COMMENT

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although FUP ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST

is the standard SPI token that begins a data list. The data list ends with ZSPI-TKN-ENDLIST.

ZFUP-TKN-FILE

is the name of the file on which the CHECKSUM command was attempted. The file name is in the Guardian internal file-name format. This token is returned with every response.

ZSPI-TKN-RETCODE

is the standard SPI return token that is returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful or an error number if an error occurred.

Table 3-1 on page 3-5 shows the ZSPI-TKN-RETCODE values specific to the CHECKSUM command

Note. Table 2-7, Errors Returned by All FUP Commands, on page 2-28 shows the ZSPI-TKN-RETCODE values common to all FUP commands.

Table 3-1. Errors Returned by CHECKSUM

Error Number	Symbolic Name (ZFUP-ERR-)	Description
52	PNAME-BAD	FUP found an invalid partition name.
53	PNAME-NOT-NET	FUP found an invalid network partition name.
68	AUDITED-FILE	An application tried a CHECKSUM on an audited file.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all of the FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages.

ZSPI-TKN-CONTEXT

is the standard SPI token that indicates whether FUP has more reply messages to return.

- If ZSPI-TKN-CONTEXT is present in the response buffer, FUP has more reply messages to return. Return the token in the command buffer to FUP. You can ignore the actual value of ZSPI-TKN-CONTEXT because this value is important only to FUP.
- If ZSPI-TKN-CONTEXT is not present in the response buffer, FUP has returned all of the reply messages.

Considerations

- CHECKSUM reads each block of data from each file specified by ZFUP-TKN-FILE and recomputes the checksum value for each block. However, CHECKSUM rewrites only the blocks with incorrect checksum values.
- Checksum errors usually indicate a potential data integrity problem. CHECKSUM recomputes the checksum value for blocks of data but does not correct any data that might have changed.
- CHECKSUM aborts if either of these occurs:
  - On a G-series system, the Subsystem Control Facility (SCF) Storage Subsystem STOP DISK or STOPOPENS DISK command was executed for the disk volume specified by ZFUP-TKN-FILE
  - On a D-series system, a Peripheral Utility Program (PUP) DOWN or STOPOPENS command was executed for the disk volume specified by ZFUP-TKN-FILE

Example

Figure 3-1 shows a TAL example of a high-level integer procedure for the CHECKSUM command:

Figure 3-1. TAL Example of a CHECKSUM Procedure (page 1 of 3)

```
!-----------------------------------------------------------
! Procedure CHECKSUM is a high-level procedure that
! demonstrates the use of the programmatic FUP interface
! with file-name templates.  RESPONSE-TYPE is set to the
! default value, and MAXRESP is set to -1.
!
! FUP returns a response record for each file and returns
! as many response records as will fit in the buffer.
! FUP will not return any more response records after it
! encounters an error.
!
! This procedure assumes that the FUP process has already
! been started.  If FUP returns an error, the procedure
! prints an error message and returns.

! This procedure returns 0 if it processes the complete
! file set or an error number if an error prevented the
! entire file set from being processed.
!-----------------------------------------------------------
INT PROC CHECKSUM (file, partonly, numberdone);

! Parameter declarations
INT     .file;       !  File(s) to checksum
INT      partonly;   !  Checksum only specified partition
INT(32) .numberdone; !  Number of files checksummed

BEGIN
! Global variables used
! STRUCT .EXT buffer (ZFUP^DDL^MSG^BUFFER^DEF);
! STRUCT .EXT save^buffer (ZFUP^DDL^MSG^BUFFER^DEF);
! INT     fup^file^number;

! Local definitions
INT     error,             ! Error variable
        spi^error,         ! SPI error variable
        done,              ! Use to process all files
        token^val,         ! Token values
```

Figure 3-1. TAL Example of a CHECKSUM Procedure (page 2 of 3)

```
        index,              ! Index variable
        number^returned;    ! Tokens returned
INT(32) token^code;         ! Token codes

! Variables to extract a map token from the buffer
INT    .checksum^par^def
           [0: (ZFUP^MAP^PAR^CHECKSUM^WLN - 1)]
           := ZFUP^MAP^PAR^CHECKSUM;
STRUCT .params (ZFUP^DDL^PAR^CHECKSUM^DEF);
STRUCT .fup^ssid (ZSPI^DDL^SSID^DEF); ! FUP SSID

numberdone := 0D;  ! Set files processed to zero

! Format buffer for CHECKSUM command


fup^ssid ':=' [ZSPI^VAL^TANDEM,ZSPI^SSN^ZFUP,
               ZFUP^VAL^VERSION];
CALL SSINIT (buffer, ZFUP^VAL^BUFLEN, fup^ssid,
             ZSPI^VAL^CMDHDR, ZFUP^CMD^CHECKSUM,
             ZFUP^OBJ^FILE);

! Put parameters into buffer

CALL SSPUTTKN (buffer, ZFUP^TKN^FILE, file);
CALL SSNULL (checksum^par^def, params);
params.zpart^only := partonly;
CALL SSPUT (buffer, checksum^par^def, params);

! Ask FUP to return as many responses as will fit in buffer

token^val := -1;
CALL SSPUTTKN (buffer, ZSPI^TKN^MAXRESP, token^val);

! Check for SPI error

CALL SSGETTKN (buffer, ZSPI^TKN^LASTERR, spi^error);
IF spi^error THEN RETURN ss^error (spi^error);

! Save the buffer for continuation requests

save^buffer ':=' buffer FOR 1 ELEMENTS;
done := FALSE;
WHILE NOT done DO
  BEGIN
  ! Send request to FUP
  error := send^to^spi^process (fup^file^number, buffer);
  IF error THEN RETURN error;
  ! Index through responses

  token^code := ZSPI^TKN^DATALIST;
  spi^error := SSGETTKN (buffer, ZSPI^TKN^COUNT,
                         token^code, 1, number^returned);
```

Figure 3-1. TAL Example of a CHECKSUM Procedure (page 3 of 3)

```
  IF spi^error THEN RETURN ss^error (spi^error);
 FOR index := 1 to number^returned DO
  BEGIN
    ! Enter first/last list
    spi^error := SSGETTKN (buffer, ZSPI^TKN^DATALIST,
                           index);
    IF spi^error THEN RETURN ss^error (spi^error);

 ! Extract response information
    spi^error:= SSGETTKN (buffer, ZSPI^TKN^RETCODE,
                            error, 1);

 IF spi^error THEN RETURN ss^error (spi^error);


    IF error THEN
      BEGIN
        CALL print^error (error);
        RETURN error;
      END
    ELSE ! no error
      numberdone := numberdone + 1D;  ! Increment counter

    ! Exit the list
    spi^error := SSGETTKN (buffer, ZSPI^TKN^ENDLIST);
    IF spi^error THEN RETURN ss^error (spi^error);
  END; ! Index through all responses in buffer

  ! Rebuild the buffer with context token

  spi^error := SSMOVETKN (ZSPI^TKN^CONTEXT, buffer,1,
                          save^buffer,1);
  IF NOT spi^error THEN
    buffer ':=' save^buffer FOR 1 ELEMENTS
  ELSE IF spi^error = ZSPI^ERR^MISTKN THEN
    done := TRUE
  ELSE
    RETURN ss^error (spi^error);
  END; !While loop
RETURN 0;  ! Return with no errors
END;       ! End of CHECKSUM procedure
```

DUPLICATE Command

The DUPLICATE command makes a copy of one or more disk files.

```
Command

ZFUP-CMD-DUPLICATE

Object Type

ZFUP-OBJ-FILE

Tokens in the Command Buffer

ZFUP-TKN-SOURCE-FILE      token-type ZSPI-TYP-FNAME.
ZFUP-TKN-DEST-FILE        token-type ZSPI-TYP-FNAME.
ZFUP-TKN-RESTART-FILE     token-type ZSPI-TYP-FNAME.

ZFUP-MAP-PAR-DUP

def ZFUP-DDL-PAR-DUP.
  02  ZPART-ONLY          type ZSPI-DDL-BOOLEAN.
  02  ZPRESERVE-TIMESTAMP type ZSPI-DDL-BOOLEAN.
  02  ZPRESERVE-OWNER     type ZSPI-DDL-BOOLEAN.
  02  ZPRESERVE-SECURITY  type ZSPI-DDL-BOOLEAN.
  02  ZDEST-OPTION        type ZSPI-DDL-ENUM.
  02  ZPRIEXT-SIZE        type ZSPI-DDL-INT2.
  02  ZSECEXT-SIZE        type ZSPI-DDL-INT2.
  02  ZDSLACK             type ZSPI-DDL-INT.
  02  ZISLACK             type ZSPI-DDL-INT.
end.

ZFUP-MAP-PART-RENAME-OPTS

def ZFUP-DDL-PART-RENAME-OPTS.
  02  ZPART-NUMBER        type ZSPI-DDL-INT.
  02  ZPART-NAME          type ZSPI-DDL-DEVICE.
  02  ZPRIEXT-SIZE        type ZSPI-DDL-INT2.
  02  ZSECEXT-SIZE        type ZSPI-DDL-INT2.
end.
ZFUP-MAP-ALT-RENAME-OPTS

def ZFUP-DDL-ALT-RENAME-OPTS.
  02 ZALT-NUMBER         type ZSPI-DDL-INT.
  02 ZALT-NAME           type ZSPI-DDL-FNAME.
end.

ZSPI-TKN-MAXRESP         token-type ZSPI-TYP-INT.
ZSPI-TKN-CONTEXT         token-type ZSPI-TYP-BYTESTRING.
ZSPI-TKN-RESPONSE-TYPE   token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT         token-type ZSPI-TYP-STRING.
```

```
Tokens in the Response Buffer

ZSPI-TKN-DATALIST        token-type ZSPI-TYP-LIST.
  ZFUP-TKN-FILE          token-type ZSPI-TYP-FNAME.
  ZSPI-TKN-RETCODE       token-type ZSPI-TYP-RETCODE.
  ZSPI-TKN-ERRLIST       token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST     token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST       token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-CONTEXT         token-type ZSPI-TYP-BYTESTRING.
```

Tokens in the Command Buffer

ZFUP-TKN-SOURCE-FILE

is a required token that specifies the file or file set to duplicate. The file name must be in the Guardian internal file-name format. Only one of these tokens is allowed per command.

If the ZFUP-TKN-RESTART-FILE token is included in the buffer, ZFUP-TKN-SOURCE-FILE must specify a single file.

ZFUP-TKN-DEST-FILE

is a required token that specifies the destination file or file set for the duplicate operation. The file name must be in the Guardian internal file-name format. Only one of these tokens is allowed per command.

Note. For more information about assigning values to ZFUP-TKN-DEST-FILE during the duplication of a file set, see Considerations on page 3-17.

ZFUP-TKN-RESTART-FILE

is an optional token that specifies the name of the restart file FUP creates to store information for the RESTART command. This token also informs FUP that the restartable option is selected. (This token is the only indication that the restartable option is selected.)

You can set ZFUP-TKN-RESTART-FILE to:

- A file name in the Guardian internal file-name format
- Blanks, which make FUP create a default restart file named ZZRSTART in your current subvolume

FUP creates the restart file (file code 855) after creating the destination file. If the duplicate operation finishes successfully, FUP purges the restart file.

ZFUP-MAP-PAR-DUP

is an optional structured token that specifies options for the DUPLICATE command. The fields are:

ZPART-ONLY

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies only the primary or secondary partitions of a partitioned file (and any nonpartitioned files) are to be duplicated, if they are designated by the source template. Entire partitioned files are not duplicated.

ZSPI-VAL-FALSE

specifies any primary partitions of a partitioned file (all partitions) are to be duplicated, if they are designated by the source template. The default is ZSPI-VAL-FALSE.

ZPRESERVE-TIMESTAMP

is a Boolean field with these values:

ZSPI-VAL-TRUE

transfers the timestamp of the source file to the destination file.

ZSPI-VAL-FALSE

specifies the timestamp of the destination file as the timestamp when the DUPLICATE command is executed. The default is ZSPI-VAL-FALSE.

ZPRESERVE-OWNER

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies that the owner ID of the source file is transferred to the destination file.

ZSPI-VAL-FALSE

specifies that the owner of the destination file is the executor of the DUPLICATE command. The default is ZSPI-VAL-FALSE.

ZPRESERVE-SECURITY

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies that the security of the source file is transferred to the destination file.

ZSPI-VAL-FALSE

specifies that the destination file assumes the default security of the executor of the DUPLICATE command. The default is ZSPI-VAL-FALSE.

ZDEST-OPTION

is an enumerated field that specifies the destination file options. The values are:

ZFUP-VAL-NEW

FUP creates a new destination file with the same characteristics as the source file. If a duplicate destination file already exists, FUP returns an error. The default is ZFUP-VAL-NEW.

ZFUP-VAL-PURGE

specifies that if the destination file already exists, it is purged before FUP duplicates the source file.

ZFUP-VAL-OLD

specifies that all of the destination files must already exist and must match the characteristics of the corresponding source files. FUP then overwrites each destination file.

ZFUP-VAL-KEEP

specifies that if a file matching the destination template already exists, the corresponding file that matches the source template is not duplicated. FUP duplicates only files that do not match the destination template.

ZFUP-MAP-PART-RENAME-OPTS

is an optional structured token that specifies options for renaming destination partitions.

These options apply only when FUP creates the destination file (when ZDEST-OPTION is either ZFUP-VAL-NEW or ZFUP-VAL-PURGE). ZPART-NUMBER and ZPART-NAME are required fields, and both must be present for each partition to be renamed. The fields are:

ZPART-NUMBER

specifies the partition file number that FUP should rename. The allowable values are in the range 1 through 15. This field does not have a default value. To rename the file, supply a value that is not null.

ZPART-NAME

specifies the new volume name for the partition specified by ZPART-NUMBER. This field does not have a default value. To rename the volume, supply a value that is not null.

ZPRIEXT-SIZE

specifies the primary extent size (in pages) of the destination file. The allowable values are in the range 0 through 512,000,000. The default is one page (2,048 bytes). If you specify an extent size over 65,535 pages, you must assign Format 2 to your files. For more information about Format 2 files, see the *File Utility Program (FUP) Reference Manual*.

ZSECEXT-SIZE

specifies the secondary extent size (in pages) of the destination partition file. The allowable values are in the range 0 through 512,000,000. The default is one page (2,048 bytes). If you specify an extent size over 65,535 pages, you must assign Format 2 to your files. For more information about Format 2 files, see the *File Utility Program (FUP) Reference Manual*.

ZFUP-MAP-ALT-RENAME-OPTS

is an optional structured token that specifies options for renaming alternate-key files in the file label of each destination file. FUP supports a maximum of 100 of these tokens.

These options apply only when FUP creates the destination file (that is, when ZDEST-OPTION is either ZFUP-VAL-NEW or ZFUP-VAL-PURGE). If this token is supplied, none of its fields can be null. The fields are:

ZALT-NUMBER

specifies the number of the alternate-key file that FUP should rename. Allowable values are in the range 0 through 255.

ZALT-NAME

specifies the new name of the alternate-key file specified by ZALT-NUMBER. This name must be in the Guardian internal file-name format.

ZSPI-TKN-MAXRESP

is the standard SPI token that indicates the maximum number of response records that FUP returns in a response buffer. The values for ZSPI-TKN-MAXRESP are:

0 FUP returns one response record per buffer (not enclosed in a data list). This is the default.

-1 FUP returns as many response records per buffer as the buffer can hold (with each response record enclosed in a data list).

n FUP returns *n* response records per buffer (with each response record enclosed in a data list). *n* must be a positive number.

ZSPI-TKN-CONTEXT

is the standard SPI token that indicates if FUP has more reply messages to return:

- If ZSPI-TKN-CONTEXT is present in the response buffer, FUP has more reply messages to return. Return the token in the command buffer to FUP. You can ignore the actual value of ZSPI-TKN-CONTEXT because this value is important only to FUP.
- If ZSPI-TKN-CONTEXT is not present in the response buffer, FUP has returned all of the reply messages.

ZSPI-TKN-RESPONSE-TYPE

is the standard SPI token that indicates the type of response records that FUP returns. The values for ZSPI-TKN-RESPONSE-TYPE are:

ZSPI-VAL-ERR-AND-WARN

FUP returns a response record only for a file that caused an error or a warning. The response buffer contains at least one error list, regardless of the value of ZSPI-TKN-RETCODE.

ZSPI-VAL-ERR-WARN-AND-NORM

FUP returns a response record for each file processed. This is the default.

ZSPI-TKN-ALLOW-TYPE

is the standard SPI token that determines whether FUP should continue processing the remaining files in a file set if an error occurs on the current file in the file set.

ZSPI-TKN-ALLOW-TYPE is significant only if ZSPI-TKN-MAXRESP is -1 (or greater than 1). Otherwise, FUP ignores the token. The values for ZSPI-TKN-ALLOW-TYPE are:

ZSPI-VAL-NORM-ONLY

FUP processes the next file only if the command was successful for the current file (that is, FUP did not return an error list for the file). This is the default.

ZSPI-VAL-WARN-AND-NORM

FUP processes the next file if the command completed for the current file. A warning might have occurred for the file, but the ZSPI-TKN-RETCODE value is zero.

ZSPI-VAL-ERR-WARN-AND-NORM

FUP processes the next file despite any problems encountered for the current file.

ZSPI-TKN-COMMENT

is the standard SPI token that lets an application include an 80-byte arbitrary comment in the command buffer. Although FUP ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST

is the standard SPI token that begins a data list. The data list ends with ZSPI-TKN-ENDLIST.

ZFUP-TKN-FILE

is the name of the file on which the command was attempted. The file name is in the Guardian internal file-name format. This token is returned with every response.

ZSPI-TKN-RETCODE

is the standard SPI return token that is returned in the response buffer by all Compaq subsystems. If the command was successful, ZSPI-TKN-RETCODE contains zero; if an error occurred, it contains an error number.

Table 3-2, Errors Returned by DUPLICATE shows the ZSPI-TKN-RETCODE values specific to the DUPLICATE command.

Note. Table 2-7, Errors Returned by All FUP Commands, on page 2-28 shows the ZSPI-TKN-RETCODE values common to all FUP commands.

Table 3-2. Errors Returned by DUPLICATE (page 1 of 2)

Error Number	Symbolic Name (ZFUP-ERR-)	Description
21	BAD-KEY	FUP found invalid alternate-key parameters.
30	AKNOUP	FUP did not change the alternate-key files.
32	DUP-SEC-PART	FUP found an invalid renames option.
33	EBCDICIN-CONFLICT	The TAPE DEFINE value for EBCDIC conflicts with ZEBCDIC.
34	SEC-PART	A secondary partition file is not allowed.
36	ENSURE-PARTS	New extent size was specified with ZPART-ONLY set to TRUE.
40	IGN-RENAME-OPTS	FUP found an invalid rename or extent size option.
41	INCOMPAT-FILE	FUP found incompatible files.
44	MISS-ALTFILE	Alternate-key file is missing.
45	MISS-PART	Partition file is missing.
48	NO-EXTSIZE	An extent size is not allowed.
50	NOT-ON-PARTF	FUP found an invalid ZDEST-OPTION for partitions.
52	PNAME-BAD	A partition name is invalid.
53	PNAME-NOT-NET	A network partition name is invalid.
54	MUST-REARRANGE-DATA	Data must be rearranged for partitions.
72	BROKEN-FILE	FUP found a broken file.
73	SAFEGUARD-LOST	Safeguard protection was lost.
76	KEPT	A file was not duplicated.
77	ALTKEY-LEN0	An alternate key has a length of zero.
78	ALTKEY-LONG	An alternate key is too long.
79	ALTFILE-PRIKEY-LONG	A primary key is too long for an alternate-key file.
80	UNIQUE-N-NON-UNIQUE	Unique and nonunique keys are mixed.
81	VARYING-UNIQUE-ALT-KEYS	Unique alternate keys are not the same length.
82	KEYLEN-ZERO	A primary key length is zero.
83	PART-KEY-MISSING	A partial key is missing.
84	ALT-KEY-MISSING	An alternate key is missing.

Table 3-2. Errors Returned by DUPLICATE (page 2 of 2)

Error Number	Symbolic Name (ZFUP-ERR-)	Description
85	ALT-FILE-MISSING	An alternate key does not have an alternate-key file.
100	REST-TOO-MANY-FILES	More than one source file is specified, and the restartable option is selected.
101	OPTICAL-RESTART-FILE	The restart file is located on an optical disk volume.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. Fpr more information about all FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages.

ZSPI-TKN-CONTEXT

is the standard SPI token that indicates whether FUP has more reply messages to return:

- If ZSPI-TKN-CONTEXT is present in the response buffer, FUP has more reply messages to return. Return the token in the command buffer to FUP. You can ignore the actual value of ZSPI-TKN-CONTEXT because this value is important only to FUP.
- If ZSPI-TKN-CONTEXT is not present in the response buffer, FUP has returned all of the reply messages.

Considerations

- When duplicating a file set, you must specify the file name part of ZFUP-TKN-DEST-FILE as an asterisk (*). Each output file is given the same file name as its corresponding input file.
- If you specify the subvolume name part of ZFUP-TKN-DEST-FILE as an asterisk, each output file is given the same subvolume name as its corresponding input file.

These considerations apply only to a DUPLICATE command with the restartable option selected:

- The restartable option is useful when you duplicate large files. If the duplicate operation fails before it completes, FUP can continue the operation from the point of failure—thus saving system time and resources.
- If a restart file is specified on an optical disk volume, or if the default subvolume of the application is on an optical disk volume (and ZFUP-TKN-RESTART-FILE contains all blanks), FUP returns the error ZFUP-ERR-OPTICAL-RESTART-FILE.

- If an application specifies an existing file for the restart file, or if the ZZRSTART file already exists (and ZFUP-TKN-RESTART-FILE contains all blanks), FUP returns the error ZFUP-ERR-FILESYS. This error indicates that a file-system error occurred. The response buffer contains a second error list indicating file-system error 10 (file already exists).

Example

Figure 3-2 shows a TAL example of a high-level procedure for the DUPLICATE command:

Figure 3-2. TAL Example of a DUPLICATE Procedure (page 1 of 3)

```
!-----------------------------------------------------------
! DUP is an example of a high-level procedure for the FUP
! DUPLICATE command.  This procedure demonstrates the use
! of the programmatic FUP interface with a file-name tem-
! plate and with ZSPI^TKN^MAXRESP, ZSPI^TKN^RESPONSE^TYPE,
! and ZSPI^TKN^ALLOW^TYPE defaulted.  FUP returns a re-
! sponse for each file in file set and one response per
! buffer.
!
! This procedure assumes that the FUP process is already
! running.  If FUP returns an error, the procedure prints
! an error message and returns.
!
! This procedure returns 0 if there are no errors or
! an error number if an error prevented the full file
! set from being processed.
!-----------------------------------------------------------

INT PROC DUP (sourcefile, destfile, numberduped, partonly,
              preserve^timestamp, preserve^owner,
              preserve^security, destoption) VARIABLE;

! Parameter Declarations
INT     .sourcefile;        ! Source file(s) to duplicate
INT     .destfile;          ! Destination
INT(32) .numberduped;       ! Number of files duplicated
INT     partonly;           ! True only if primary
!                             partition
INT     preserve^timestamp; ! Save timestamp (Boolean)
INT     preserve^owner;     ! Save ownership (Boolean)
INT     preserve^security;  ! Save security (Boolean)
INT     destoption;         ! Disposition of destination
```

Figure 3-2. TAL Example of a DUPLICATE Procedure (page 2 of 3)

```
BEGIN
! Global variables used
! STRUCT .EXT buffer (ZFUP^DDL^MSG^BUFFER^DEF);
! STRUCT .EXT save^buffer (ZFUP^DDL^MSG^BUFFER^DEF);
! INT     fup^file^number;
! Local Definitions
INT     error, spi^error, done;
INT    .dup^par^def
            [0:(ZFUP^MAP^PAR^DUP^WLN - 1)]
            := ZFUP^MAP^PAR^DUP;
STRUCT .params (ZFUP^DDL^PAR^DUP^DEF);
STRUCT .fup^ssid (ZSPI^DDL^SSID^DEF);

! Check for required parameters
IF NOT $PARAM (sourcefile) OR NOT $PARAM (destfile) OR
   NOT $PARAM (numberduped) THEN
  RETURN an^error;numberduped := 0D;    ! Set to zero

! Format buffer for DUPLICATE command

fup^ssid ':=' [ZSPI^VAL^TANDEM,ZSPI^SSN^ZFUP,
               ZFUP^VAL^VERSION];
CALL SSINIT (buffer, ZFUP^VAL^BUFLEN, fup^ssid,
                     ZSPI^VAL^CMDHDR, ZFUP^CMD^DUPLICATE,
                     ZFUP^OBJ^FILE);

! Put required parameters into buffer
CALL SSPUTTKN (buffer, ZFUP^TKN^SOURCE^FILE, sourcefile);
CALL SSPUTTKN (buffer, ZFUP^TKN^DEST^FILE, destfile);

! Process optional parameters
IF $PARAM (partonly) OR $PARAM (preserve^timestamp) OR
   $PARAM (preserve^owner) OR $PARAM (preserve^security) OR
   $PARAM (destoption) THEN


IF spi^error THEN RETURN ss^error (spi^error);

! Save the buffer for continuation requests
save^buffer ':=' buffer FOR 1 ELEMENTS;

done := FALSE;
WHILE NOT done DO
```

Figure 3-2. TAL Example of a DUPLICATE Procedure (page 3 of 3)

```
BEGIN
  CALL SSNULL (dup^par^def, params);
  IF $PARAM (partonly) THEN
    params.zpart^only := partonly;
  IF $PARAM (preserve^timestamp ) THEN
    params.zpreserve^timestamp := preserve^timestamp;
  IF $PARAM (preserve^owner ) THEN
    params.zpreserve^owner := preserve^owner;
  IF $PARAM (preserve^security ) THEN
    params.zpreserve^security := preserve^security;
  IF $PARAM (destoption ) THEN
    params.zdest^option := destoption;
  CALL SSPUT (buffer, dup^par^def, params);
  END;! Check for SPI error
CALL SSGETTKN (buffer, ZSPI^TKN^LASTERR, spi^error);IF error
THEN
    BEGIN
    CALL print^error (error);
    RETURN error;
  ELSE
    numberduped := numberduped + 1D;    ! Increment by 1

! Rebuild the buffer with context token
  spi^error := SSMOVETKN (ZSPI^TKN^CONTEXT,buffer,1,
                          save^buffer,1);
  IF NOT spi^error THEN
    buffer ':=' save^buffer FOR 1 ELEMENTS
  ELSE IF spi^error = ZSPI^ERR^MISTKN THEN
    done := TRUE
  ELSE
    RETURN ss^error (spi^error);
  END; ! WHILE loop
  RETURN 0;
END; ! DUPLICATE Procedure
```

GETVERSION Command

The GETVERSION command returns the FUP server version in the ZSPI-TKN-SERVER-VERSION token (in the SPI header) and the server ID string in the token ZSPI-TKN-SERVER-BANNER.

```
Command

ZFUP-CMD-GETVERSION

Object Type

None.

Tokens in the Command Buffer

ZSPI-TKN-COMMENT          token-type ZSPI-TYP-STRING.

Tokens in the Response Buffer

ZSPI-TKN-SERVER-BANNER    token-type ZSPI-TYP-CHAR50.
ZSPI-TKN-RETCODE          token-type ZSPI-TYP-RETCODE.

ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ...
  ZSPI-TKN-ENDLIST        token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZSPI-TKN-COMMENT

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although FUP ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-SERVER-BANNER

is a 50-character string that contains the version ID of the FUP server, the FUP release date, and the FUP compilation date as follows:

FUP - T9074*vff* - *release-date* - *compilation-date*

where:

vff

is the FUP version ID. An example is D40.

`release-date`

is the FUP release date. An example is 01APR01.

`compilation-date`

is the FUP compilation date. An example of this is 01APR97.

`ZSPI-TKN-RETCODE`

is the standard SPI return token that is returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful, or an error number if an error occurred.

Note. Table 2-7, Errors Returned by All FUP Commands, on page 2-28 shows the ZSPI-TKN-RETCODE values common to all FUP commands.

`ZSPI-TKN-ERRLIST`

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages.

LOAD Command

The LOAD command loads data into a structured disk file without affecting any associated alternate-key files.

```
Command

ZFUP-CMD-LOAD

Object Type

ZFUP-OBJ-FILE

Tokens in the Command Buffer

ZFUP-TKN-SOURCE-FILE         token-type ZSPI-TYP-FNAME.

ZFUP-TKN-DEST-FILE           token-type ZSPI-TYP-FNAME.

ZFUP-MAP-LOAD-SOURCE-OPTS

def ZFUP-DDL-LOAD-SOURCE-OPTS.
  02  ZBLOCK-SIZE            type ZSPI-DDL-INT2.
  02  ZRECORD-SIZE           type ZSPI-DDL-INT2.
  02  ZVAR-REC               type ZSPI-DDL-BOOLEAN.
  02  ZSHARE                 type ZSPI-DDL-BOOLEAN.
  02  ZEMPTYOK               type ZSPI-DDL-BOOLEAN.
  02  ZEBCDIC                type ZSPI-DDL-BOOLEAN.
  02  ZUNLOAD                type ZSPI-DDL-BOOLEAN.
  02  ZREWIND                type ZSPI-DDL-BOOLEAN.
  02  ZSKIP                  type ZSPI-DDL-INT.
  02  ZREELS                 type ZSPI-DDL-INT.
  02  ZMOUNT-MSG-FILE        type ZSPI-DDL-FNAME.
  02  ZTRIMCHAR-IS-PRESENT   type ZSPI-DDL-BOOLEAN.
  02  ZTRIMCHAR              type ZSPI-DDL-CHAR.
end.

ZFUP-MAP-LOAD-DEST-OPTS

def ZFUP-DDL-LOAD-DEST-OPTS.
  02  ZCOMPACT               type ZSPI-DDL-BOOLEAN.
  02  ZPADCHAR-IS-PRESENT    type ZSPI-DDL-BOOLEAN.
  02  ZPADCHAR               type ZSPI-DDL-CHAR.
end.

ZFUP-MAP-LOAD-KEYSEQ-OPTS

def ZFUP-DDL-LOAD-KEYSEQ-OPTS.
```

```
  02  ZPARTOF                 type ZSPI-DDL-DEVICE.
  02  ZSORTED                type ZSPI-DDL-BOOLEAN.
  02  ZMAX-RECS              type ZSPI-DDL-INT4.
  02  ZSCRATCH               type ZSPI-DDL-FNAME.
  02  ZDSLACK                type ZSPI-DDL-INT.
  02  ZISLACK                type ZSPI-DDL-INT.
end.


ZSPI-TKN-MAXRESP            token-type ZSPI-TYP-INT.
ZSPI-TKN-RESPONSE-TYPE      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE         token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT            token-type ZSPI-TYP-STRING.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST       token-type ZSPI-TYP-LIST.
  ZFUP-TKN-FILE         token-type ZSPI-TYP-FNAME.
  ZSPI-TKN-RETCODE      token-type ZSPI-TYP-RETCODE.

  ZFUP-MAP-LOAD-XFER-COUNTS

  def ZFUP-DDL-LOAD-XFER-CNTS.
    02  ZNAME            type ZSPI-DDL-FNAME.
    02  ZCOUNT           type ZSPI-DDL-INT4.
  end.

  ZSPI-TKN-ERRLIST      token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST    token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST      token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZFUP-TKN-SOURCE-FILE

is a required token that specifies the file containing the records to be loaded. The file name must be in the Guardian internal file-name format. Only one of these tokens is allowed per command.

ZFUP-TKN-DEST-FILE

is a required token that specifies an existing disk file to which FUP loads the records from the source file. The file name must be in the Guardian internal file-name format. Only one of these tokens is allowed per command.

ZFUP-MAP-LOAD-SOURCE-OPTS

is an optional structured token that specifies options for the source file. The fields are:

ZBLOCK-SIZE

specifies the number of bytes in an input block. The allowable values are in the range 1 through 32,767.

ZRECORD-SIZE

specifies the maximum number of bytes in an input record. The allowable values are in the range 1 through 4096.

ZVAR-REC

is a Boolean field with these values:

ZSPI-VAL-TRUE

FUP reads variable-length, blocked records.

ZSPI-VAL-FALSE

FUP does not read variable-length, blocked records. The default is ZSPI-VAL-FALSE.

ZSHARE

is a Boolean field that applies only to disk files. It can have these values:

ZSPI-VAL-TRUE

FUP opens the source file with the shared exclusion mode.

ZSPI-VAL-FALSE

FUP opens the source file with protected exclusion mode. The default is ZSPI-VAL-FALSE.

ZEMPTYOK

is a Boolean field with these values:

ZSPI-VAL-TRUE

FUP disables the check for an empty source file.

ZSPI-VAL-FALSE

FUP returns an error for an empty source file. The default is ZSPI-VAL-FALSE.

ZEBCDIC

is a Boolean field with these values:

ZSPI-VAL-TRUE

FUP assumes that the source file contains EBCDIC characters and translates the characters to their ASCII equivalents.

ZSPI-VAL-FALSE

FUP assumes that the source file contains ASCII characters. The default is ZSPI-VAL-FALSE.

ZUNLOAD

is a Boolean field that applies only to magnetic tape. It can have these values:

ZSPI-VAL-TRUE

FUP unloads the tape after rewinding. The default is ZSPI-VAL-TRUE.

ZSPI-VAL-FALSE

FUP does not unload the tape after rewinding.

ZREWIND

is a Boolean field that applies only to magnetic tape. It can have these values:

ZSPI-VAL-TRUE

FUP rewinds the tape at the end of file. The default is ZSPI-VAL-TRUE.

ZSPI-VAL-FALSE

FUP does not rewind the tape at the end of file.

ZSKIP

is a field that applies only to magnetic tape; it specifies the number of end-of-file marks that FUP should skip before beginning the data transfer. The allowable values are -255 through 255. The default is 0.

ZREELS

is a field that applies only to magnetic tape; it specifies the number of tape reels for the source file. The allowable values are 1 through 255. The default is 1.

ZMOUNT-MSG-FILE

is a field that applies only to magnetic tape; it specifies the name of a disk file that contains a tape mount message if a tape reel must be mounted by an operator. If this name is not supplied and a mount request is required, FUP returns an error.

ZTRIMCHAR-IS-PRESENT

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies that the trim character (ZTRIMCHAR) is supplied. You must use this field if a value is supplied in ZTRIMCHAR.

ZSPI-VAL-FALSE

specifies that the trim character (ZTRIMCHAR) is not supplied. This is the default.

ZTRIMCHAR

specifies an ASCII trim character. Any trailing characters matching this character are deleted from the input record. You must specify ZSPI-VAL-TRUE with ZTRIMCHAR-IS-PRESENT to supply this field.

ZFUP-MAP-LOAD-DEST-OPTS

is an optional structured token that specifies options for the destination file. The fields are:

ZCOMPACT

is a Boolean field that applies only to relative source files. It can have these values:

ZSPI-VAL-TRUE

FUP ignores records with a length of zero and does not represent them in the destination file. This is the default.

ZSPI-VAL-FALSE

FUP represents records with a length of zero in the destination file.

ZPADCHAR-IS-PRESENT

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies that the pad character field (ZPADCHAR) is supplied. This is the default. You must use this field if a value is supplied in ZPADCHAR.

ZSPI-VAL-FALSE

specifies that the pad character field (ZPADCHAR) is not supplied.

ZPADCHAR

is a field that specifies an ASCII pad character. If an input file record length contains fewer bytes than the output file record length, FUP pads the destination file record with this character (up to the output file record length). To supply this field, specify ZSPI-VAL-TRUE with ZPADCHAR-IS-PRESENT.

ZFUP-MAP-LOAD-KEYSEQ-OPTS

is an optional structured token that specifies options for a key-sequenced destination file. Only one of these tokens is allowed per command. The fields are:

ZPARTOF

specifies the volume where the primary partition of the destination file resides. Only the partition designated by the destination file name is to be loaded.

ZSORTED

is a Boolean field with these values:

ZSPI-VAL-TRUE

specifies that the records in the source file are in the key field order of the destination file and should not be sorted.

ZSPI-VAL-FALSE

specifies that the source file records should be sorted. The default is ZSPI-VAL-FALSE.

ZMAX-RECS

is an estimate of the number of records in the source file. The allowable values are in the range 0 through 2,147,483,647. The default is 10,000.

ZSCRATCH

specifies a file name or volume name to be used for temporary storage during the sorting process. The name must be in the Guardian internal file-name format. The default is a scratch file on the default volume.

ZDSLACK

specifies the minimum percentage of slack space to be left for future insertions in data blocks. The allowable values are in the range 0 through 99. The default is 0.

ZISLACK

specifies the minimum percentage of slack space to be left for future insertions in index blocks. The allowable values are in the range 0 through 99. The default is 0.

ZSPI-TKN-MAXRESP, ZSPI-TKN-RESPONSE-TYPE, ZSPI-TKN-ALLOW-TYPE

are standard SPI response-control tokens. If you can set these tokens, they have no effect on the execution of the command.

ZSPI-TKN-COMMENT

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although FUP ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZFUP-TKN-FILE

is the name of the file on which the command was attempted. The file name is in the Guardian internal file-name format. This token is returned with every response.

ZSPI-TKN-RETCODE

is the standard SPI return token that is returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful or an error number if an error occurred.

Table 3-3 shows the ZSPI-TKN-RETCODE values specific to the LOAD command .

Note. Table 2-7, Errors Returned by All FUP Commands, on page 2-28 shows the ZSPI-TKN-RETCODE values common to all FUP commands.

Table 3-3. Errors Returned by LOAD (page 1 of 2)

Error Number	Symbolic Name (ZFUP-ERR-)	Description
22	BAD-PARTS	FUP detected invalid partition parameters.
23	BAD-TAPELABEL	FUP detected an invalid tape label.
24	BAD-VAR-BLOCKLEN	FUP detected an invalid block length for a variable-length record.
25	BAD-VAR-RECLEN	FUP detected an invalid variable-length record.
26	BLOCKIN-CONFLICT	FUP detected a conflict with the block length.
27	BLOCKLEN-BIG	FUP detected a block length that is too large.
28	DEFINE-CONFLICT	FUP encountered a TAPE DEFINE conflict with ZBLOCK-SIZE or ZRECORD-SIZE.
33	EBCDICIN-CONFLICT	FUP encountered a TAPE DEFINE conflict with ZEBCDIC.
35	EMPTY-SOURCE	The source file is empty.
38	IGN-COMPACT	FUP ignored the ZCOMPACT option.
42	INCON-PARTS	FUP detected inconsistent partition files.

Table 3-3. Errors Returned by LOAD (page 2 of 2)

Error Number	Symbolic Name (ZFUP-ERR-)	Description
43	INV-FTYPE	FUP could not load an unstructured file.
55	RECIN-CONFLICT	FUP encountered a TAPE DEFINE conflict with the record length.
56	RECLEN-BIG	A TAPE DEFINE RECLEN is too large.
62	TRUNC	Truncation is occurring.
64	USE-EXT-N-READ	FUP encountered a TAPE DEFINE USE EXTEND error.
65	USE-OUT-N-READ	FUP encountered a TAPE DEFINE USE OUT error.
66	VAR-TRUNC	Truncation occurred on the last variable-length record.
69	SKIPIN-CONFLICT	FUP encountered a TAPE DEFINE conflict with ZSKIP.
70	REELS-CONFLICT	FUP encountered a TAPE DEFINE conflict with ZREELS.
75	NO-DEFINE	A TAPE DEFINE was not found.

ZFUP-MAP-LOAD-XFER-CNTS

is a structured token that specifies the transfer counts for the LOAD command. The fields are:

ZNAME

specifies the name of the file or partition loaded.

ZCOUNT

specifies the number of records loaded into the file or partition identified by ZNAME.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages.

Considerations

- If the loaded file is not partitioned, FUP returns only one ZFUP-MAP-LOAD-XFER-CNTS structure. Otherwise, FUP returns one structure for each partition loaded.

- If you specify the ZPARTOF option when loading a nonpartitioned file, FUP returns an error message. If the base partition specified with ZPARTOF does not exist, file-system error #11 (file not in directory) is returned. If the base partition specified with ZPARTOF is not a partitioned file (or is not a partition of the file being loaded), FUP returns the ZFUP-ERR-BAD-PARTS error list.
- To load a secondary partition, you must specify the name of the secondary partition as ZFUP-TKN-DEST-FILE, and specify the name of the volume where the primary partition resides for the ZPARTOF option.
- To load only the primary partition, specify the name of the primary partition as ZFUP-TKN-DEST-FILE, and specify the name of the primary volume for the ZPARTOF option.
- If the ZFUP-TKN-SOURCE-FILE records are sorted, disk space for the sort scratch file and for ZFUP-TKN-DEST-FILE must exist concurrently during the sorting phase.
- Data that contains the ZPADCHAR or ZTRIMCHAR character can be altered or lost. An example of this occurs if you pad each record in a data file with zeros to a specific size in bytes and store the records in another file. If you then trim the trailing zeros, any original data ending with zero is trimmed. To avoid this problem, use a ZPADCHAR or ZTRIMCHAR character that is not part of your data.
- The ZCOMPACT option affects only relative files. If you set ZCOMPACT to ZSPI-VAL-TRUE to load data from a nonrelative file, FUP returns the warning error list ZFUP-ERR-IGN-COMPACT.
- If the input file is a relative file that contains zero-length (empty) records, and ZCOMPACT is set to ZSPI-VAL-TRUE, FUP ignores the zero-length records. The remaining records (other than the zero-length records) are compacted at the beginning of the file. If you are loading relative files, always set ZCOMPACT to ZSPI-VAL-FALSE.
- When alternate-key records are not built because the full alternate key does not exist within the primary record, FUP returns the ZFUP-ERR-SHORT-KEYS error list.

Example

See the LOADALTFILE command Example on page 3-37.

LOADALTFILE Command

The LOADALTFILE command generates alternate-key records from a specified primary file and loads the alternate-key records into an alternate-key file.

```
Command

ZFUP-CMD-LOADALTFILE

Object Type

ZFUP-OBJ-FILE

Tokens in the Command Buffer

ZFUP-TKN-SOURCE-FILE          token-type ZSPI-TYP-FNAME.
ZFUP-TKN-ALTFILE-NUM          token-type ZSPI-TYP-INT.
ZFUP-MAP-PAR-LOADALTFILE

def ZFUP-DDL-PAR-LOADALTFILE.
  02  ZMAX-RECS               type ZSPI-DDL-INT4.
  02  ZSCRATCH                type ZSPI-DDL-FNAME.
  02  ZDSLACK                 type ZSPI-DDL-INT.
  02  ZISLACK                 type ZSPI-DDL-INT.
end.

ZSPI-TKN-MAXRESP              token-type ZSPI-TYP-INT.
ZSPI-TKN-RESPONSE-TYPE        token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE           token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT              token-type ZSPI-TYP-STRING.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST             token-type ZSPI-TYP-LIST.

  ZFUP-TKN-FILE               token-type ZSPI-TYP-FNAME.

  ZFUP-MAP-LOADALT-XFER-CNTS

  def ZFUP-DDL-LOADALT-XFER-CNTS.
    02  ZNAME                 type ZSPI-DDL-FNAME.
    02  ZCOUNT                type ZSPI-DDL-INT4.
  end.

  ZFUP-TKN-READ-COUNT         token-type ZSPI-TYP-INT4.
  ZSPI-TKN-RETCODE            token-type ZSPI-TYP-RETCODE.
  ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZFUP-TKN-SOURCE-FILE

is a required token that specifies an existing primary file whose alternate-key records are to be generated and loaded into the file indicated by the alternate-key file number (ZFUP-TKN-ALTFILE-NUM token). The file name must be in the Guardian internal file-name format. Only one of these tokens is allowed per command.

ZFUP-TKN-ALTFILE-NUM

is a required token that specifies the alternate-key file number of the file that FUP should load. The allowable values are in the range 0 through 255. Only one of these tokens is allowed per command.

ZFUP-MAP-PAR-LOADALTFILE

is an optional structured token that specifies LOADALTFILE options. The fields are:

ZMAX-RECS

is an estimate of the number of records in the source file. FUP multiplies this value by the number of alternate keys associated with the alternate-key file to determine the size of the scratch file used by the FASTSORT process. The allowable values are in the range 0 through 2,147,483,647. This value does not have to be exact, but it should be greater than (or equal to) the number of records in the source file. The default is 10,000.

ZSCRATCH

specifies a file name or volume name to be used for temporary storage during the FASTSORT process. The name must be in Guardian internal file-name format. The default is a scratch file on the default volume.

ZDSLACK

specifies the minimum percentage of slack space to be left in the data blocks of the destination. Allowable values are in the range 0 through 99. The default is 0.

ZISLACK

specifies the minimum percentage of slack space to be left in the index blocks of the destination. Allowable values are in the range 0 through 99. The default is 0.

ZSPI-TKN-MAXRESP, ZSPI-TKN-RESPONSE-TYPE, ZSPI-TKN-ALLOW-TYPE

are standard SPI response-control tokens. If you set these tokens, they have no effect on the execution of the command.

ZSPI-TKN-COMMENT

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although FUP ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST

is the standard SPI token that begins a data list. The data list ends with ZSPI-TKN-ENDLIST.

ZFUP-TKN-FILE

is the name of the primary file from which the LOADALTFILE command reads records. The file name is in the Guardian internal file-name format. This token is returned with every response.

ZFUP-MAP-LOADALT-XFER-CNTS

is a structured token that specifies the transfer counts for the LOADALTFILE command. The fields are:

ZNAME

specifies the name of the file or partition loaded.

ZCOUNT

specifies the number of records loaded into the file or partition identified by ZNAME.

ZFUP-TKN-READ-COUNT

specifies the number of primary records read.

ZSPI-TKN-RETCODE

is the standard SPI return token returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful, or an error number if an error occurred.

Table 3-5, Errors Returned by RESTART, on page 3-40 shows the ZSPI-TKN-RETCODE values specific to the LOADALTFILE command.

Note. Table 2-7, Errors Returned by All FUP Commands, on page 2-28 shows the ZSPI-TKN-RETCODE values common to all FUP commands.

Table 3-4. Errors Returned by LOADALTFILE

Error Number	Symbolic Name (ZFUP-ERR-)	Description
21	BAD-KEY	FUP detected invalid alternate-key parameters.
37	FILE-KEY-INCOM	An alternate-key file is invalid with the specified alternate keys.
47	NO-ALT-FILE	An alternate-key file does not exist.
59	SHORT-KEYS	FUP detected incomplete alternate-key fields.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages

Considerations

- FUP returns one structure for each partition loaded, but it returns only one ZFUP-MAP-LOADALT-XFER-CNTS structure if the loaded file is not partitioned.
- To perform a LOADALTFILE operation, you must have both read and write access to both the primary and alternate-key files.
- LOADALTFILE ignores any NO UPDATE specifications, and it honors a NULL specification that is defined for a key field. (FUP does not generate an alternate-key record for a field that has only null characters if the null character is defined.)
- A sort operation occurs as you execute LOADALTFILE. The primary file is read sequentially with its primary-key field. For each record read from the primary file, the alternate-key records are generated and written to the FASTSORT process.
- When the sort completes, the sorted records are read from the FASTSORT process and loaded into the indicated alternate-key file. During the sorting phase, disk space for the sort scratch file and the alternate-key file must exist concurrently.
- A LOADALTFILE operation fails a duplicate key in an alternate-key file has the UNIQUE attribute. If the attributes of the alternate-key file are incorrect, LOADALTFILE fails and FUP returns the ZFUP-ERR-FILE-KEY-INCOM error list.
- LOADALTFILE does not generate alternate-key file records or return an explanatory error list in these cases:
  - The full length of the alternate-key field is not contained in the primary record.
  - A null value was specified for the key, and the field contains only the null value.

Example

Figure 3-3, TAL Example of a LOADALTFILE Procedure shows a TAL example of a high-level integer procedure for the LOADALTFILE command.

Figure 3-3. TAL Example of a LOADALTFILE Procedure (page 1 of 2)

```
!-----------------------------------------------------------
! LOADALTFILE is an example of a high-level procedure
! for the FUP LOADALTFILE command.  This procedure assumes
! that the FUP process has already been started.
!
! This procedure returns 0 if successful, or an error
! number otherwise.
!-----------------------------------------------------------

INT PROC LOADALTFILE (altno, prifile, maxrecs, scratch,
                      dslack, islack) VARIABLE;

! Parameter Declarations

INT   altno;      ! Alternate file number to load
INT   .prifile;   ! Primary file to read
FIXED maxrecs;    ! Estimate of maximum number
                    of records
INT   .scratch;   ! Scratch file name
INT   dslack;     ! Data slack percentage
INT   islack;     ! Index slack percentage

BEGIN
! Global variables used
! STRUCT .EXT buffer (ZFUP^DDL^MSG^BUFFER^DEF);
! INT     fup^file^number;

! Local Definitions
INT      error,
         spi^error;
INT     .loadaltfile^par^def
            [0: (ZFUP^MAP^PAR^LOADALTFILE^WLN-1)]
            := ZFUP^MAP^PAR^LOADALTFILE;
```

Figure 3-3. TAL Example of a LOADALTFILE Procedure (page 2 of 2)

```
STRUCT  .params (ZFUP^DDL^PAR^LOADALTFILE^DEF);
STRUCT  .fup^ssid (ZSPI^DDL^SSID^DEF);

! Check for required parameters

IF NOT $PARAM (altno) OR NOT $PARAM (prifile) THEN
  RETURN an^error;

! Format buffer for LOADALTFILE command

fup^ssid ':=' [ZSPI^VAL^TANDEM,ZSPI^SSN^ZFUP,
               ZFUP^VAL^VERSION];
CALL SSINIT (buffer, ZFUP^VAL^BUFLEN, fup^ssid,
             ZSPI^VAL^CMDHDR, ZFUP^CMD^LOADALTFILE,
             ZFUP^OBJ^FILE);

! Put required parameters into buffer
CALL SSPUTTKN (buffer, ZFUP^TKN^SOURCE^FILE, prifile);
CALL SSPUTTKN (buffer, ZFUP^TKN^ALTFILE^NUM, altno);

! Process optional parameters
IF $PARAM (maxrecs) OR $PARAM (scratch)
   OR $PARAM (dslack) OR $PARAM( islack ) THEN
    BEGIN
    CALL SSNULL (loadaltfile^par^def, params);

    IF $PARAM (maxrecs ) THEN
       params.zmax^recs := maxrecs;
    IF $PARAM (Scratch ) THEN
       params.zscratch ':=' scratch FOR 24 BYTES;
    IF $PARAM (dslack ) THEN
       params.zdslack := dslack;
    IF $PARAM (islack ) THEN
       params.zislack := islack;

    CALL SSPUT (buffer, loadaltfile^par^def, params);
    END;

! Check for SPI error
CALL SSGETTKN (buffer, ZSPI^TKN^LASTERR, spi^error);
IF spi^error THEN RETURN ss^error (spi^error);

! Send request to FUP
error := send^to^spi^process (fup^file^number, buffer);
IF error THEN RETURN error;

! Interpret the response
spi^error := SSGETTKN (buffer, ZSPI^TKN^RETCODE,
                       error, 1 !index!);
IF spi^error THEN RETURN ss^error (spi^error);
IF error THEN RETURN error;
RETURN 0;
END; ! LOADALTFILE Procedure
```

RESTART Command

The RESTART command continues a duplication operation for a DUPLICATE command that failed before it was finished. The DUPLICATE command must have been executed with the restartable option specified. RESTART uses information stored in a disk restart file to determine where the restart should begin.

```
Command

ZFUP-CMD-RESTART

Object Type

ZFUP-OBJ-FILE

Tokens in the Command Buffer

ZFUP-TKN-RESTART-FILE    token-type ZSPI-TYP-FNAME.

ZSPI-TKN-MAXRESP         token-type ZSPI-TYP-INT.
ZSPI-TKN-RESPONSE-TYPE   token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE      token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT         token-type ZSPI-TYP-STRING.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST        token-type ZSPI-TYP-LIST.
  ZFUP-TKN-FILE          token-type ZSPI-TYP-FNAME.
  ZSPI-TKN-RETCODE       token-type ZSPI-TYP-RETCODE.
  ZSPI-TKN-ERRLIST       token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST     token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST       token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZFUP-TKN-RESTART-FILE

is an optional token that specifies the name of the restart file. The file name must be in the Guardian internal file-name format. Only one of these tokens is allowed per command.

The restart file is an unstructured disk file created by the DUPLICATE command. It must have a file code of 855.

If ZFUP-TKN-RESTART-FILE contains all blanks (or is not present in the command buffer), FUP searches for a restart file named ZZRSTART on the current subvolume of the application.

ZSPI-TKN-MAXRESP, ZSPI-TKN-RESPONSE-TYPE, ZSPI-TKN-ALLOW-TYPE

are standard SPI response-control tokens. If you set these tokens, they have no effect on the execution of the command.

ZSPI-TKN-COMMENT

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although FUP ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST

is the standard SPI token that begins a data list. The data list ends with ZSPI-TKN-ENDLIST.

ZFUP-TKN-FILE

is the name of the restart file specified in the command buffer. The file name is in the Guardian internal file-name format.

ZFUP-TKN-RETCODE

is the standard SPI return token returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful or an error number if an error occurred.

Table 3-5, Errors Returned by RESTART shows the ZSPI-TKN-RETCODE values specific to the RESTART command.

Note. Table 2-7, Errors Returned by All FUP Commands, on page 2-28 shows the ZSPI-TKN-RETCODE values common to all FUP commands.

Table 3-5. Errors Returned by RESTART

Error Number	Symbolic Name (ZFUP-ERR-)	Description
101	OPTICAL-RESTART-FILE	The restart file is on an optical disk volume.
103	SRC-FILE-CHANGED	The source file was changed.
104	DEST-NOT-CORRUPT	The destination file is not corrupt.
105	INFO-INVALID	The restart file contains invalid information.
106	DP-CHANGED	The disk-process format was changed for either the source or the destination file disk volume.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all FUP error numbers and their corresponding error lists, see Appendix B, FUP Error Messages.

Considerations

- When the RESTART command is issued, FUP verifies:
  - The destination file is corrupt.
  - The restart file has a file code of 855.
  - The restart file contains valid information.
  - The source file has not been modified since the original DUPLICATE command was issued. FUP checks the time of the last modification in the file label.
  - The restart file does not reside on an optical disk volume (if the restart file is explicitly specified).
  - The current subvolume of the application is not on an optical disk volume if the restart file is not specified (if ZFUP-TKN-RESTART-FILE contains all blanks or is not present in the command buffer).
- If any of these checks fail, FUP terminates the RESTART command and returns an error list in the response buffer.
- Except for the time of the last modification, FUP does not check the other attributes of the source file. Do not change an attribute that might affect a restart operation. For example, if you change the security attributes for the source file, a restart might not be possible.
- After the restart, FUP continues to update the restart file. If the duplicate operation fails again, a subsequent restart can continue from the second point of failure.

4

ORSERV Programmatic Interface

This section describes how a management application program uses the Subsystem Programmatic Interface (SPI) to communicate with an ORSERV process. You can use SPI to execute these ORSERV commands from an application written in C, COBOL85, TACL or TAL:

GETVERSION	Returns the ORSERV server version and the server ID
ONLINERELOAD	Performs an online reload of a key-sequenced file or SQL object
STATUS	Returns the status of an online reload operation (either executing or suspended)
SUSPEND	Suspends an executing online reload operation.

Note. Before you read this section, you should be familiar with the information in the *SPI Programming Manual*.

Communicating With ORSERV

To communicate with an ORSERV process, you must follow a few basic procedures. These procedures are listed in Figure 4-1 on page 4-2 and described here using the same numbering:

1. Start an ORSERV process.

   You must start ORSERV on the same system as the target file. If you start ORSERV on one system to reload a file on another system, you receive error 10 (ZORS-ERR-WRONG-SERVER).

   You can use a process-creation procedure (such as PROCESS_LAUNCH_) to start an ORSERV process. You can find ORSERV in the $SYSTEM.SYS*nn*.ORSERV program file.

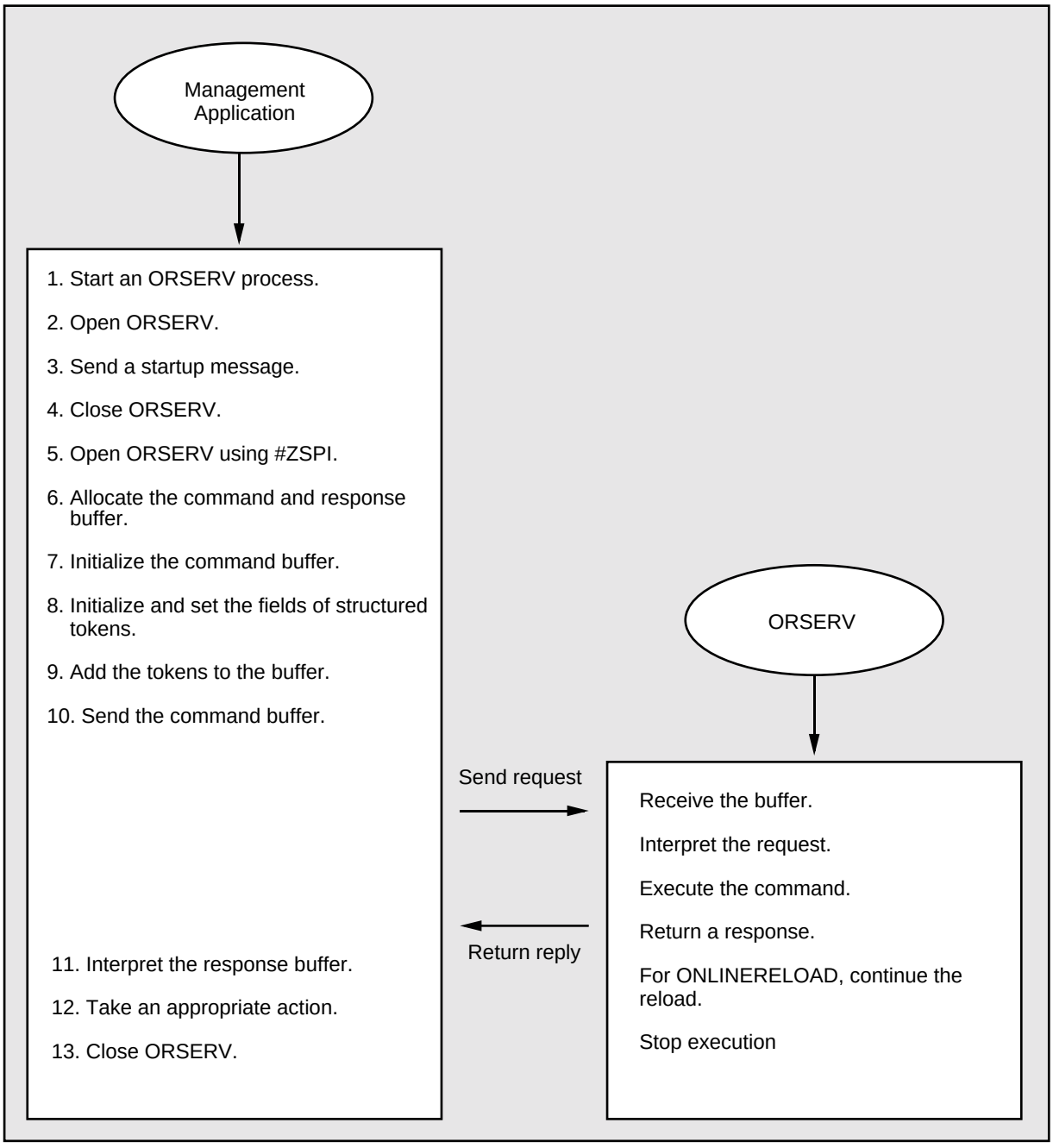
   Before you start the ORSERV process, use the CREATEPROCESSNAME procedure to obtain a system-assigned process name.

   Note. For more system procedure information, see the *Guardian Procedure Calls Reference Manual*.

2. Open ORSERV.

   The process file name must have the internal file-name format. You can include backup OPENs and CLOSEs in ORSERV.

Figure 4-1. Communicating With ORSERV



When you open the ORSERV server:

- You can open ORSERV only if you created the ORSERV process.

- You can have only one ORSERV process open.
- You must not open ORSERV using NOWAIT I/O.
- The SYNC depth cannot be greater than one.

Check for any file-system errors after you open ORSERV. File-system errors associated with applications using SPI include:

- (Error 12) Your management process tried to open ORSERV more than once.
- (Error 17) A problem occurred during the backup OPEN. This can occur when the backup OPEN does not have a matching primary OPEN or if the backup and primary OPEN parameters do not match.
- (Error 28) The SYNC or NOWAIT depth is greater than one.
- (Error 48) ORSERV rejects the OPEN because your user ID does not have the proper authority to perform the OPEN. This can occur when the process trying to open ORSERV is not the creator of the ORSERV process.

Note. For a description of each file-system error, see the *Guardian Procedure Errors and Messages Manual.*

---

3. Send ORSERV a startup message.

   When ORSERV reads the startup message, it ignores the IN file, the OUT file, the default volume and subvolume, and the text portion. It also ignores the AUTOSTOP parameter and any ASSIGN or PARAM messages.

4. Close ORSERV after you send the startup message.

5. Open ORSERV with #ZSPI as the first qualifier (in the process file name). The first qualifier of the process file name for a backup OPEN must also be #ZSPI. If the process name qualifier is not #ZSPI, you receive file-system error 11.

6. Allocate the command and response buffer.

   The recommended minimum buffer length is ZORS-VAL-BUFLEN. A buffer of this length is large enough to hold the command and response for all ORSERV programmatic commands.

Note. Although this manual refers to the SSINIT, SSNULL, SSGET, SSPUT, and SSMOVE procedures, Compaq also provides the SSPUTTKN, SSGETTKN, and SSMOVETKN procedures. These procedures have the same functions as SSPUT, SSGET, and SSMOVE, and a calling sequence that is simpler for TAL programming.

For TACL programming, Compaq also provides the #SSINIT, #SSNULL, #SSGET, #SSGETV, #SSPUT, #SSPUTV, and #SSMOVE built-in functions.

---

7. Initialize the command buffer.

   Call the SSINIT procedure to specify the ORSERV command and initialize the buffer (including the addition of header tokens).

8. Initialize and set the fields of structured tokens.

   Call the SSNULL procedure to initialize the fields of each extensible structured token to null values, then set the individual fields of each structured token.

9. Add the tokens to the buffer.

   Call the SSPUT procedure for each token that you want put in the buffer. Specify the command buffer, the unique token code, and a token value for each token. SSPUT places the token values in the buffer.

10. Send the buffer.

    Send the command buffer to ORSERV with the procedure that is appropriate for the language you are using (such as WRITEREAD for TAL, READ WITH PROMPT for COBOL85, or an #APPENDV/#REPLYV loop for TACL).

    Although ORSERV accepts only one ONLINERELOAD, STATUS, or SUSPEND command from an application, it can execute a GETVERSION command before it uses one of them.

    Check for any file-system errors after you send the command buffer to ORSERV. File-system errors associated with applications using SPI include:

    Error 2 The request is not a correctly formatted SPI buffer. This can happen if the first two bytes do not contain -28.

    Error 60 You did not open ORSERV before you sent the command buffer.

    After ORSERV receives the buffer, it interprets the command request, executes the command (if no errors exist in the command format), and returns a response in the buffer (including any execution errors).

11. Interpret the response buffer from ORSERV.

    After ORSERV returns the response buffer, call the SSGET procedure to retrieve the tokens (including any error tokens) from the buffer. You must call SSGET to get each token in the buffer.

12. Take the appropriate action.

    Decide what to do after checking the ORSERV reply in the response buffer. If ORSERV returned an error, execute an error-handling routine.

13. Close ORSERV.

    Call the CLOSE file-system procedure to stop communicating with ORSERV. ORSERV stops after you close it, unless it is executing a RELOAD, in which case it stops after it completes the RELOAD. Before you close ORSERV, consider:

    - ONLINERELOAD command

      The reload operation is started, and it returns the response buffer when you send an ONLINERELOAD command. If reload errors occur, ORSERV writes them to the ZZRELOAD file. To read the ZZRELOAD file and check the status

of the reload operation, use the STATUS command. The ORSERV process stops after it completes the RELOAD.

Note. ORSERV and the target file must be on the same system, or you receive an error message (ZORS-ERR-WRONG-SERVER).

- STATUS and SUSPEND commands

  The response buffer is returned, and stops after ORSERV executes a STATUS or SUSPEND command.

Starting and Opening ORSERV (TAL Example)

Figure 4-2 shows an example of a TAL integer procedure that is used to start and open an ORSERV process for programmatic use.

Figure 4-2. TAL Procedure to Start and Open ORSERV (page 1 of 2)

```
! ----------------------------------------------------------
! Procedure START^AND^OPEN^ORSERV starts a new ORSERV
! process and opens it for programmatic use.  If an error
! occurs, the procedure calls an error-handling routine.
! ----------------------------------------------------------
PROC start^and^open^orserv;
BEGIN
INT .ORSERV^program^file^name [0:11] :=
                              ["$SYSTEM SYSTEM  ORSERV  "];
! ----------------------------------------------------------
! Set the startup message ID to -1; ORSERV ignores the
! remaining fields in the startup message.
! ----------------------------------------------------------
startup^message.message^id := -1;
! ----------------------------------------------------------
! Call PROCESS_LAUNCH_ to create the ORSERV process.  First,
! call CREATEPROCESSNAME to get the ORSERV process name.
! The ORSERV^process^name qualifier must be #ZSPI.
! ----------------------------------------------------------
ORSERV^process^name ':=' "                              ";
CALL CREATEPROCESSNAME (ORSERV^process^name);
CALL PROCESS_LAUNCH_ (ORSERV^program^file^name,
                 ! priority !,
                 ! memory^pages !,
                 ! processor !,
                 ! process^id !,
                 error,
                 ORSERV^process^name);
! Check for error; call error routine if error occurs.
IF error THEN CALL newprocess^error^handler (error);
```

Figure 4-2. TAL Procedure to Start and Open ORSERV (page 2 of 2)

```
! ---------------------------------------------------------
! Open the ORSERV process and check for an error.
! ---------------------------------------------------------
CALL OPEN (ORSERV^process^name,
           ORSERV^file^number);
IF <> THEN
  BEGIN
  CALL FILEINFO (ORSERV^file^number,
                 error);

IF error THEN CALL fs^error^handler (error);
  END;

! ---------------------------------------------------------
! Send the startup message and check for an error.
! ---------------------------------------------------------
CALL WRITE (ORSERV^file^number,
            startup^message,
            $OFFSET (startup^message.text));
IF <> THEN
  BEGIN
  CALL FILEINFO (ORSERV^file^number,
                 error);
  IF error THEN CALL fs^error^handler (error);
  END;
CALL CLOSE (ORSERV^file^number);
ORSERV^process^name [4] ':=' "#ZSPI           ";
CALL OPEN (ORSERV^process^name,
           ORSERV^file^number);
IF <> THEN
  BEGIN
  CALL FILEINFO (ORSERV^file^number,
                 error);
  IF error THEN CALL fs^error^handler (error);
  END;
END;      ! of START^AND^OPEN^ORSERV procedure
```

Sending a Buffer to ORSERV (TAL Example)

Figure 4-3, TAL Procedure to Send a Buffer to ORSERV shows an example of a TAL integer procedure that sends and receives an ORSERV command and response buffer using the WRITEREAD procedure.

This example searches for a file-system error, verifies that it is an SPI buffer, and checks if the entire buffer was read.

Note. For more information on creating checks, see Receiving and Decoding a Response Buffer on page 4-13.

Figure 4-3. TAL Procedure to Send a Buffer to ORSERV (page 1 of 2)

```
! ---------------------------------------------------------
! Procedure SEND^COMMAND sends a command buffer to ORSERV
! using the WRITEREAD procedure.  It then receives and
! checks the response buffer.  The ORSERV process must
! already be created (with PROCESS_LAUNCH_ [NOWAIT]) and open.
! The buffer length must be a minimum ZORS^VAL^BUFLEN
! bytes. This procedure returns 0 if no errors occur, or
! an error number, if an error occurs.
! ---------------------------------------------------------
INT PROC send^command;
! ---------------------------------------------------------
! Global variables used are:
!   STRUCT .buffer (ZORS^DDL^MSG^BUFFER^DEF);
!   INT     ORSERV^file^number,
!           error, parameter^error, spi^error;
! ---------------------------------------------------------
BEGIN
! ---------------------------------------------------------
! Local definitions.
! ---------------------------------------------------------
INT .buffer^header (ZSPI^DDL^HEADER^DEF) = buffer;
INT  used^length,
     initial^buffer^position,

LITERAL buffer^error = 99;
! ---------------------------------------------------------
! Get the used length value and send the buffer to ORSERV.
! Then, check for a file-system error.
! ---------------------------------------------------------
CALL SSGETTKN (buffer,
               ZSPI^TKN^USEDLEN,
               used^length);
```

Figure 4-3. TAL Procedure to Send a Buffer to ORSERV (page 2 of 2)

```
CALL WRITEREAD (ORSERV^file^number,
                buffer,
                used^length,
                ZORS^VAL^BUFLEN);

IF < THEN
  BEGIN
  CALL FILEINFO(ORSERV^file^number,error);

  IF error THEN CALL fs^error^handler (error);
  END;
! ----------------------------------------------------------
! Check that the response buffer is an SPI buffer
! and that ORSERV read all of the buffer.
! ----------------------------------------------------------

IF buffer^header.z^msgcode <> -28 THEN
  RETURN buffer^error;
CALL SSGETTKN (buffer,
               ZSPI^TKN^USEDLEN,
               used^length);
IF used^length > ZORS^VAL^BUFLEN THEN
  RETURN buffer^error;
! ----------------------------------------------------------
! Reset the length, position, and last error.
! ----------------------------------------------------------
buffer^header.Z^BUFLEN := ZORS^VAL^BUFLEN;
initial^buffer^position := ZSPI^VAL^INITIAL^BUFFER;
CALL SSPUTTKN (buffer,
               ZSPI^TKN^INITIAL^POSITION,
               initial^buffer^position);
CALL SSPUTTKN (buffer, ZSPI^TKN^CLEARERR);
RETURN 0; ! Successful return
END;      ! of SEND^COMMAND procedure.
```

Elements of SPI Messages for ORSERV

A command and response buffer contains special codes called tokens. Each token contains a specific piece of information such as the ORSERV command number or object type.

For example, ZORS-CMD-ONLINERELOAD is the command number token for the ONLINERELOAD command, and ZORS-OBJ-FILE is the object type token for a file.

Source Definition Files

When you write your application, each module must include the SPI standard definitions and the ORSERV definitions. Depending on the language you are using, include these files with your source code (the disk volume is selected at your site):

C	Use the #INCLUDE directive to include the ZSPIDEF.ZSPIC and ZSPIDEF.ZORSC files.
COBOL85	Use the COPY statement to include ZSPIDEF.ZSPICOB and ZSPIDEF.ZORSCOB. Use the COPY statement with the REPLACING option to include a section of a file.

TACL	Load the ZSPIDEF.ZSPITACL and ZSPIDEF.ZORSTACL files. To avoid buffer overflows during loading, load each file:
	`PUSH X`
	`#LOAD / LOADED X / $`*`volume`*`.ZSPIDEF.ZORSTACL`
	`POP X`

TAL	Use the ?SOURCE directive to include ZSPIDEF.ZSPITAL and ZSPIDEF.ZORSTAL files.

Naming Rules for Applications

Compaq uses names beginning with the letter Z for all definitions and all fields of structures in definition files. To avoid conflicts with names defined by Compaq, do not begin any names you define in your application with an uppercase or lowercase Z.

Common ORSERV Command Syntax Elements

Command Numbers

Each ORSERV command is assigned a unique command number. A command number is represented by a symbolic name in the form ZORS-CMD-*name* (*name* identifies the command). For example, the symbolic name for the STATUS command is ZORS-CMD-STATUS. Table 4-1 on page 4-10 show the ORSERV programmatic commands and object types.

Object Types

The object type designates the object of each command. Each ORSERV command (except GETVERSION) requires the ZORS-OBJ-FILE object type. GETVERSION requires the ZORS-OBJ-NULL object type.

Note. For more information about object names, see SPI Programming Considerations for ORSERV on page 4-12.

Table 4-1. ORSERV Commands and Object Types

Symbolic Name (ZORS-CMD-)	Object Type (ZORS-OBJ-)
GETVERSION	NULL
ONLINERELOAD	FILE
STATUS	FILE
SUSPEND	FILE

Additional ORSERV Command and Response Buffer Tokens

These paragraphs contain additional tokens that are used in ORSERV command or response buffers—including simple token codes, token types, predefined value names, field types, token maps and structured tokens.

Simple Token Codes

These are represented by symbolic names using the form Z*sss*-TKN-*name*, where *sss* is the subsystem abbreviation and *name* is the token code. An example of a simple token code for ORSERV is ZORS-TKN-FILE. This identifies the target file for a command.

Note. For more information about the token codes defined by ORSERV, see Section 5, ORSERV Commands and Responses.

Token Types

These are represented by symbolic names using the form Z*sss*-TYP-*name*, where *sss* is the subsystem abbreviation and *name* is the token type. ORSERV does not define any private token types (such as token types with the name ZORS-TYP-*name*). Table 4-2, Standard SPI Token Types Used by ORSERV, on page 4-11 shows the standard SPI token types used by ORSERV.

Predefined Value Names

These are represented by symbolic names using the form Z*sss*-VAL-*name*, where *sss* is the subsystem abbreviation and *name* is the predefined value. An example of a predefined value name for ORSERV is ZORS-VAL-BUFLEN. The recommended buffer length for the ORSERV command and response buffer.

Token Maps and Structured Tokens

A token map is a variable-length integer array that contains decoding information and a reference name for an extensible structured token. A token map contains a token code and a description of the token value—including the token fields, the null values for the fields, and the version number for the fields. An application uses a token map to pass information in an extensible structured token to ORSERV.

Note. For more information about ORSERV token maps and structured tokens, see Section 5, ORSERV Commands and Responses.

Field Types

ORSERV uses standard SPI field types that are represented using the form ZSPI-DDL-*name* (*name* specifies the field type). An example of this is the message buffer ZORS-DDL-MSG-BUFFER.

Event Messages

Although ORSERV does not report event messages to the Event Management Service (EMS), event messages might be generated by the NonStop Kernel or the disk process when you use the programmatic interface to ORSERV.

Note. For more information on event messages, see the *Operator Messages Manual*.

Standard SPI Token Types

Table 4-2. Standard SPI Token Types Used by ORSERV (page 1 of 2)

Token Type	Description
ZSPI-TYP-BOOLEAN	16-bit signed Boolean value
ZSPI-TYP-BYTE	8-bit unsigned integer
ZSPI-TYP-BYTESTRING	String of 8-bit unsigned integers
ZSPI-TYP-CHAR	8-bit ASCII character
ZSPI-TYP-CHAR-PAIR	Pair of 8-bit ASCII characters
ZSPI-TYP-CHAR8	Eight 8-bit ASCII characters
ZSPI-TYP-CHAR24	Twenty-four 8-bit ASCII characters
ZSPI-TYP-CHAR50	Fifty 8-bit ASCII characters
ZSPI-TYP-DEVICE	8-byte internal device name
ZSPI-TYP-ENUM	16-bit signed enumerated value
ZSPI-TYP-ERROR	SPI error token
ZSPI-TYP-FLT2	64-bit floating-point number
ZSPI-TYP-FNAME	24-byte internal file name
ZSPI-TYP-INT	16-bit signed integer

Table 4-2. Standard SPI Token Types Used by ORSERV (page 2 of 2)

Token Type	Description
ZSPI-TYP-INT2	32-bit signed integer
ZSPI-TYP-INT2-PAIR	Pair of 32-bit signed integers
ZSPI-TYP-INT4	64-bit fixed-point number
ZSPI-TYP-LIST	Token starting a list
ZSPI-TYP-POSITION	64-bit SPI position descriptor
ZSPI-TYP-SSCTL	Special SPI control operation
ZSPI-TYP-SSID	Subsystem ID
ZSPI-TYP-STRING	Variable-length ASCII string
ZSPI-TYP-SUBVOL	16-byte internal subvolume name
ZSPI-TYP-UINT	16-bit unsigned integer

SPI Programming Considerations for ORSERV

The *SPI Programming Manual* provides programming considerations for management applications that use SPI command and response buffers to communicate with subsystems such as ORSERV. This subsection describes considerations specific to ORSERV.

Building the Command Buffer

Your application must allocate a command and response buffer to communicate with ORSERV. This buffer must be large enough to hold each ORSERV command and response. The recommended buffer length in bytes is ZORS-VAL-BUFLEN.

Your application uses SPI procedures to initialize and add tokens to the command buffer. To initialize the buffer and to specify the ORSERV command, use the SSINIT procedure. (Multiple ORSERV commands per buffer are not supported.) SSINIT also adds the header tokens to the buffer.

When the buffer is initialized with SSINIT, any previous contents of the buffer are overwritten. Therefore, to save the contents of the buffer before you call SSINIT, use the SSMOVE procedure to save the contents in a separate variable.

To initialize the fields of extensible structured tokens to null values, use SSNULL. Your application should always call SSNULL, even if it explicitly sets all currently defined fields of the token. This ensures that your application continues to run correctly if future versions of ORSERV add fields to these structured tokens.

Specifying an Object Name

Each ORSERV programmatic command (except GETVERSION) requires an object type of ZORS-OBJ-FILE. GETVERSION requires the ZORS-OBJ-NULL object type.

ORSERV accepts a single file name as the object of a command. The name cannot be a file set or file-set list.

You specify the target file name in the ZORS-TKN-FILE token. The file name must be in the Guardian internal file-name format and must be fully qualified. If the file is not on the same system as ORSERV, the name must be in Guardian network internal file-name format. If a file name does not include a system identifier, ORSERV assumes that the file is on the same system that ORSERV is running. All file names returned by ORSERV in the response buffer are also in the Guardian internal file-name format.

Note. For more information about the Guardian internal file-name format, see the *Guardian Programmer's Guide*.

Discontinuing a Command in Progress

ORSERV does not check for CANCEL requests from an application that executes the CANCEL file-system procedure. You must start another ORSERV process and send it a SUSPEND command to suspend a reload operation.

Note. For more information, see the description of the SUSPEND Command on page 5-18.

Receiving and Decoding a Response Buffer

When your application receives the response buffer, first check for these errors and take an appropriate action (if an error occurs):

- Errors that were reported by the method used to send and receive the buffer, such as a file-system error
- ZSPI-TKN-RETCODE token for a value indicating that ORSERV found an error in the command
- Used-length token ZSPI-TKN-USEDLEN from the response buffer to ensure that the buffer is not larger than the buffer allocated by your application

You might also want to check the values of these tokens in the message header to ensure that the response matches the original command:

ZSPI-TKN-COMMAND	ORSERV command number from the original request
ZSPI-TKN-OBJECT-TYPE	ORSERV object type number from the original request
ZSPI-TKN-SSID	Subsystem ID of the ORSERV server that performed the command
ZSPI-TKN-SERVER-VERSION	Release version of the ORSERV server that performed the command

After these checks are made, you can extract the remaining tokens from the buffer and continue processing.

Extracting Tokens From an ORSERV Response Record

An ORSERV command generates a single response record in the response buffer. This response record describes the action of the ORSERV command on a single file. Figure 4-4 shows a single ORSERV response record. The tokens are not necessarily in the same order as the ones displayed here.

Figure 4-4. Tokens in a Response Buffer

```
token-1
   token-2
      .
      .
      .
   token-n
   ZSPI-TKN-RETCODE
   ZORS-TKN-FILE
   error list, if an error or warning occurs
```

The tokens in a single response record are:

- ORSERV response tokens

  These tokens (token-1, token-2, through token-*n*) are specific to the ORSERV command. For example, the STATUS command returns the ZORS-MAP-STATUS-RESPONSE token.

- ZSPI-TKN-RETCODE token

  This token identifies any errors or warnings. A value of zero indicates that the command was successful. A value other than zero indicates that an error or warning occurred. A value of zero for the ONLINERELOAD command indicates that the reload operation was initiated successfully.

- ZORS-TKN-FILE token

  This token identifies the object of the ORSERV command (the target file).

- Error lists

  If any errors or warnings occurred, the error or warning information appears in one or more error lists.

Handling ORSERV Errors

If ZSPI-TKN-RETCODE indicates an error or warning (a value other than zero), the response buffer contains one or more error lists. For the values that ZSPI-TKN-RETCODE can return, see:

- The errors common to all ORSERV commands in Table 4-4 on page 4-21
- The individual ORSERV command descriptions, which display the errors that each command can return

- All of the ORSERV errors (including a cause and recovery for each error) in Appendix C, ORSERV Error Messages.

An error list begins with ZSPI-TKN-ERRLIST and ends with ZSPI-TKN-ENDLIST. Each error list describes an error or warning. The ZSPI-TKN-ERROR error token, which is included in the error list, specifies the error that occurred. Other tokens in the error list describe other components of the error, such as the ORSERV command and the file ORSERV was processing when the error occurred.

If ZSPI-TKN-RETCODE is zero, the response buffer can still contain an error list. This error list is a warning describing a condition you might want to know about—although the condition did not prevent ORSERV from performing the requested command. To determine the warning, enter the error list and check the value of the ZSPI-TKN-ERROR token.

Figure 4-5 shows an example of an ORSERV error list. The tokens are not necessarily in the same order as the ones displayed here.

Figure 4-5. Tokens in an Error List

```
ZSPI-TKN-ERRLIST
   ZSPI-TKN-ERROR
   token-1
   token-2
      .
      .
      .
   token-n
   nested error list, if another subsystem error occurs
ZSPI-TKN-ENDLIST
```

The tokens in the error list are:

- ZSPI-TKN-ERROR

  This token contains the ORSERV subsystem ID and the error number. ZSPI-TKN-ERROR is always present in an error list.

- Error-description tokens

  These optional tokens (token-1, token-2, through token-*n*) describe the error.

- Nested error lists

  ORSERV returns a second nested error list when the error originates from another subsystem or software component (such as the Guardian file system).

Types of ORSERV Errors

These types of errors can occur when your application sends a command buffer to ORSERV:

- Syntax errors in the ORSERV command format

- Command failure errors encountered by ORSERV
- Command failure errors encountered by a subsystem or software component other than ORSERV

Syntax Errors in the Command Format

ORSERV first evaluates the command format in the buffer to determine if any syntax errors exist. Examples are an invalid token value or an invalid command.

The ORSERV error list contains the ZSPI-TKN-ERROR token, which contains the error number and the ORSERV subsystem ID, and the ZSPI-TKN-PARM-ERR token, which identifies the token or field that caused the error. Because the command failed before ORSERV started the reload operation, the error list does not contain a file-name token.

An example of an invalid token value error occurs if a field within a structured token is not within its valid range. ZSPI-TKN-RETCODE has a value of ZORS-ERR-INV-VALUE (7) to indicate this error.

Figure 4-6. Error List for an Invalid Token Value

```
ZSPI-TKN-ERRLIST
  ZSPI-TKN-ERROR
     Z-SSID          ! ORSERV subsystem ID
     Z-ERROR         ! ZORS-ERR-INV-VALUE (7)
  ZSPI-TKN-PARM-ERR
     Z-TOKENCODE     ! Token code that caused the error
     Z-INDEX         ! Occurrence number of the token
     Z-OFFSET        ! Byte offset of the field
ZSPI-TKN-ENDLIST
```

ORSERV Errors

These errors occur when ORSERV tries to execute the command. For example, error 15 (ZORS-ERR-ORELOAD-INPROGRESS) occurs when ORSERV attempts an ONLINERELOAD command for a target file that already has a reload operation in progress.

The error list contains the token ZSPI-TKN-ERROR and a token map ZORS-MAP-CMD-ERROR, which identifies the command that failed, the object type, and the file that ORSERV was processing when the error occurred.

Figure 4-7. Error List for an ORSERV Command Failure

```
ZSPI-TKN-ERRLIST
  ZSPI-TKN-ERROR
     Z-SSID           ! ORSERV subsystem ID
     Z-ERROR          ! ZORS-ERR-ORELOAD-INPROGRESS (15)
  ZORS-MAP-CMD-ERROR
     ZCOMMAND         ! ZORS-CMD-ONLINERELOAD command
     ZOBJECT          ! Object type (ZORS-OBJ-FILE)
     ZNAME            ! Name of the target file
ZSPI-TKN-ENDLIST
```

Other Subsystem and Software Component Errors

The NonStop Kernel and the Guardian file system are sometimes called to perform various tasks when ORSERV executes a command.

Errors can originate from these subsystems or software components. When this type of error occurs, ZSPI-TKN-RETCODE indicates that the command failed with an error from a source other than ORSERV. The ORSERV error list contains the token ZSPI-TKN-ERROR, the token map ZORS-MAP-CMD-ERROR, and a nested error list describing the actual error.

If the ONLINERELOAD command failed because of a file-system error on an open procedure call, the ZSPI-TKN-RETCODE token has the value ZORS-ERR-FILESYS (error 13).

Figure 4-8. Example of an ORSERV Nested Error List

```
ZSPI-TKN-ERRLIST     ! Start of ORSERV error list
  ZSPI-TKN-ERROR
    Z-SSID           ! ORSERV subsystem ID
    Z-ERROR          ! ZORS-ERR-FILESYS (13)
  ZORS-MAP-CMD-ERROR
    ZCOMMAND         ! ZORS-CMD-ONLINERELOAD command (1)
    ZOBJECT          ! ZORS-OBJ-FILE object type (1)
    ZNAME            ! File that ORSERV was processing

  ZSPI-TKN-ERRLIST       ! Start of file-system error list
    ZSPI-TKN-ERROR
      Z-SSID             ! File-system subsystem ID
      Z-ERROR            ! File-system error number
    ZSPI-TKN-PROC-ERR    ! Open procedure call
  ZSPI-TKN-ENDLIST       ! End of file-system error list
ZSPI-TKN-ENDLIST         ! End of ORSERV error list
```

The TAL example in Figure 4-9 on page 4-18 demonstrates how a sequence of SSGET calls can extract the tokens from a nested error list—although the error checks and error-handling routines are omitted from the example.

Figure 4-9. Extracting Tokens From an ORSERV Nested Error List

```
! Get ZSPI^TKN^RETCODE from the response buffer.
SSGETTKN (buffer,
          ZSPI^TKN^RETCODE,
          return^code, 1)


! ------------------------------------ ORSERV error list
! Enter the ORSERV error list.
SSGETTKN (buffer,
          ZSPI^TKN^ERRLIST, 1);
! Get the ORSERV error number and SSID.
SSGETTKN (buffer,
          ZSPI^TKN^ERROR,
          orserv^error,, 1);
! Get the ORSERV command error information
SSGET (buffer,
       ZORS^MAP^CMD^ERROR,
       error^info, 1)
! ------------------------------- File-system error list
!   Enter the file-system error list.
    SSGETTKN (buffer,
              ZSPI^TKN^ERRLIST, 1)
!   Get file-system error number and SSID.
    SSGETTKN (buffer,
              ZSPI^TKN^ERROR,
              filesys^error, 1)
!   Get the file-system procedure that caused the error.
    SSGETTKN (buffer,
              ZSPI^TKN^PROC^ERR,
              filesys^procedure, 1)
!   Exit the file-system error error list.
    SSGETTKN (buffer,
              ZSPI^TKN^ENDLIST)
! Exit the ORSERV error list.
SSGETTKN (buffer,
          ZSPI^TKN^ENDLIST)
```

Common Definitions

The ORSERV programmatic commands use SPI standard definitions and ORSERV definitions. Although this subsection provides a general description of these definitions, for specific information about each definition, see Section 5, ORSERV Commands and Responses, or Appendix C, ORSERV Error Messages.

Note. All definitions are shown in DDL (or COBOL85) format using hyphens (-) as separators.

- If you are programming in TAL or TACL, substitute the circumflex (^) symbol for the hyphens.
- If you are programming in C, substitute the underscore (_) symbol for the hyphens.

SPI Standard Definitions

SPI standard definitions begin with ZSPI- and appear in the source definition files ZSPITAL, ZSPICOB, ZSPIPAS, ZSPIC, and ZSPITACL.

Note. For more information, see Source Definition Files on page 4-9.

Table 4-3. SPI Standard Definitions Used by ORSERV (page 1 of 2)

Header Tokens		
ZSPI-TKN-CHECKSUM	ZSPI-TKN-LASTPOSITION	ZSPI-TKN-SERVER-VERSION
ZSPI-TKN-COMMAND	ZSPI-TKN-MAX-FIELD-VERSION	ZSPI-TKN-SSID
ZSPI-TKN-HDRTYPE	ZSPI-TKN-MAXRESP	ZSPI-TKN-USEDLEN
ZSPI-TKN-LASTERR	ZSPI-TKN-OBJECT-TYPE	
ZSPI-TKN-LASTERRCODE	ZSPI-TKN-POSITION	
Special Tokens		
ZSPI-TKN-ADDR	ZSPI-TKN-INITIAL-POSITION	ZSPI-TKN-OFFSET
ZSPI-TKN-CLEARERR	ZSPI-TKN-LEN	ZSPI-TKN-RESET-BUFFER
ZSPI-TKN-COUNT	ZSPI-TKN-NEXTCODE	
ZSPI-TKN-DEFAULT-SSID	ZSPI-TKN-NEXTTOKEN	
Other Simple Tokens		
ZSPI-TKN-ALLOW-TYPE	ZSPI-TKN-ENDLIST	ZSPI-TKN-RESPONSE-TYPE
ZSPI-TKN-COMMENT	ZSPI-TKN-ERRLIST	ZSPI-TKN-RETCODE
ZSPI-TKN-CONTEXT	ZSPI-TKN-ERROR	ZSPI-TKN-SERVER-BANNER
ZSPI-TKN-DATALIST	ZSPI-TKN-PARM-ERR	
Value Names		
ZSPI-SSN-ZORS	ZSPI-VAL-FALSE	ZSPI-VAL-TANDEM
ZSPI-VAL-TRUE		

Table 4-3. SPI Standard Definitions Used by ORSERV (page 2 of 2)

Token Types		
ZSPI-TYP-BOOLEAN	ZSPI-TYP-ENUM	ZSPI-TYP-SSCTL
ZSPI-TYP-BYTE-PAIR	ZSPI-TYP-ERROR	ZSPI-TYP-SSID
ZSPI-TYP-BYTESTRING	ZSPI-TYP-FNAME32	ZSPI-TYP-STRING
ZSPI-TYP-CHAR8	ZSPI-TYP-INT	ZSPI-TYP-TIMESTAMP
ZSPI-TYP-CHAR50	ZSPI-TYP-LIST	ZSPI-TYP-UINT
ZSPI-TYP-CRTPID	ZSPI-TYP-MARK	
ZSPI-TYP-DEVICE	ZSPI-TYP-PARM-ERR	
Structures		
ZSPI-DDL-BOOLEAN	ZSPI-DDL-ENUM	ZSPI-DDL-INT2
ZSPI-DDL-BYTE	ZSPI-DDL-FNAME	ZSPI-DDL-TIMESTAMP
ZSPI-DDL-CHAR8	ZSPI-DDL-FNAME32	ZSPI-DDL-UINT
ZSPI-DDL-CRTPID	ZSPI-DDL-INT	
ZSPI-DDL-DEVICE	ZSPI-DDL-INT-PAIR	

Information about the SPI standard definitions specific to ORSERV is provided here:

Note. Each SPI standard definition is in the *SPI Programming Manual*.

ZSPI-SSN-ZORS

is the subsystem number assigned to ORSERV.

ZSPI-TKN-COMMAND

contains the command number for these ORSERV programmatic commands—GETVERSION, ONLINERELOAD, STATUS, or SUSPEND. Command numbers and their associated commands appear later in this section.

ZSPI-TKN-ERROR

is the error token that is present in an error list. This token contains the ORSERV subsystem ID and the ORSERV error number.

Note. For more information about all of the ORSERV numbers and their associated error lists, see Appendix C, ORSERV Error Messages.

ZSPI-TKN-PARM-ERR

is a parameter error token present in some error lists. This token identifies a token code (Z-TOKENCODE field), the occurrence number of the token (Z-INDEX field), and the byte offset of a specific field in the token (Z-OFFSET field).

ZSPI-TKN-OBJECT-TYPE

contains the object-type number for the ORSERV object. The object-type for ORSERV commands is ZORS-OBJ-FILE (except for the GETVERSION command, which is ZORS-OBJ-NULL).

ZSPI-TKN-RETCODE

is the return token. Table 4-4 shows the ZSPI-TKN-RETCODE values common to all ORSERV commands:

Table 4-4. Errors Returned by All ORSERV Commands

Error Number	Symbolic Name (ZORS-ERR-)	Description
0	OK	The command completed successfully.
1	INV-COMMAND	ORSERV found an invalid command.
2	INV-OBJECT	ORSERV found an invalid object type.
3	INVALID-TOKEN	ORSERV found an invalid token.
4	MISS-TOKEN	ORSERV detected a missing token.
5	MISS-FIELD	ORSERV detected a missing field in a structured token.
6	EXTRA-TOKEN	ORSERV found an extra token.
7	INV-VALUE	ORSERV found an invalid token or field value.
8	LONG-COMMAND	A command was too long for the buffer.
9	WRONG-SSID	The application specified an invalid ORSERV subsystem ID.
10	WRONG-SERVER	The application specified an invalid ORSERV server.
11	SPI	An SPI subsystem error occurred.
12	PE	A programming error occurred.
13	FILESYS	A file-system error occurred.
14	GUARD	A NonStop Kernel error occurred.

ZSPI-TKN-SSID

contains ZORS-VAL-SSID, the subsystem ID of the ORSERV subsystem. ZORS-VAL-SSID has this structure:

```
def ZORS-VAL-SSID tacl ssid.
   02 Z-FILLER            type character 8
                          value is ZSPI-VAL-TANDEM.
   02 Z-OWNER             type ZSPI-DDL-CHAR8
                          redefines Z-FILLER.
   02 Z-NUMBER            type ZSPI-DDL-INT
                          value is ZSPI-SSN-ZORS.
   02 Z-VERSION           type ZSPI-DDL-UINT
                          value is ZORS-VAL-VERSION.
end.
```

ZSPI-TKN-SERVER-VERSION

contains the server version of the ORSERV subsystem.

ORSERV Definitions

ORSERV definitions begin with ZORS- and appear in the source definition files ZORSTAL, ZORSCOB, ZORSTACL, and ZORSC. The ORSERV definitions are described here.

Note. For more information about the ORSERV definitions, see Section 5, ORSERV Commands and Responses.

ORSERV Message Buffer Declaration

ZORS-DDL-MSG-BUFFER

is the SPI buffer you use for ORSERV commands. An example of this buffer is:

```
def ZORS-DDL-MSG-BUFFER.
   02 Z-MSGCODE            type ZSPI-DDL-INT.
   02 Z-BUFLEN             type ZSPI-DDL-UINT.
   02 Z-OCCURS             type ZSPI-DDL-UINT.
   02 Z-FILLER             type ZSPI-DDL-BYTE
                           occurs 0 TO ZORS-VAL-BUFLEN
times
                           depending on Z-OCCURS.
end.
```

Predefined Token Values

ZORS-VAL-BUFLEN

is the recommended buffer length (in bytes) for all ORSERV command and response buffers.

ZORS-VAL-BUFLEN-W

is the buffer length in words.

ZORS-VAL-SSID

is the ORSERV subsystem ID. The definition for ZORS-VAL-SSID appears in ZSPI-TKN-SERVER-VERSION on page 4-22.

ZORS-VAL-VERSION

is the release version number of the ORSERV subsystem.

Simple and Structured Tokens

These simple tokens are specific to ORSERV:

ZORS-TKN-FILE

specifies the name of the target file that is the object of the reload operation.

ZORS-TKN-VOLUME

specifies the name of a disk volume.

Note. For definitions of the structured tokens specific to ORSERV, see Section 5, ORSERV Commands and Responses.

Tokens in Error Lists

This token is used in ORSERV error lists:

ZORS-MAP-CMD-ERROR

describes a command error and contains these fields:

ZNAME

is the name of the file that ORSERV was processing (or attempting to process) when the error occurred.

ZCOMMAND

is the ORSERV command that was executing when the error occurred.

ZOBJECT

is the object type for the ORSERV command. The object type is always ZORS-OBJ-FILE for the ONLINERELOAD, STATUS, and SUSPEND commands. The object type is ZORS-OBJ-NULL for the GETVERSION command.

Note. For more information about all ORSERV error numbers and their corresponding error lists, see Appendix C, ORSERV Error Messages.

5

ORSERV Commands and Responses

This section describes the ORSERV programmatic commands and responses for:

Each description contains:

- A header showing the command name
- A summary of the function of the command
- A box that lists these elements for each command:
  - The symbolic name for the command number
  - The symbolic name of the object type
  - A list of tokens that can be used in the command buffer
  - A list of tokens that ORSERV can return in the response buffer
- A description of tokens listed in the box
- Considerations for using the command
- An example of the command

While reading the descriptions, consider:

- Although the list of the tokens in the box is not necessarily in the order that the tokens will actually appear in a command or response buffer, the token ZSPI-TKN-ENDLIST always appears at the end of a list started by ZSPI-TKN-DATALIST or ZSPI-TKN-ERRLIST.
- The notation used in the box for simple tokens is a shortened version of the DDL TOKEN-CODE statement. Structured tokens are defined using the DDL DEF statement.

GETVERSION Command

The GETVERSION command returns the ORSERV server version in the ZSPI-TKN-SERVER-VERSION token (in the SPI header) and the server ID string in the ZSPI-TKN-SERVER-BANNER token.

```
Command

ZORS-CMD-GETVERSION

Object Type

ZORS-OBJ-NULL

Tokens in the Command Buffer

ZSPI-TKN-COMMENT          token-type ZSPI-TYP-STRING.

Tokens in the Response Buffer

ZSPI-TKN-SERVER-BANNER    token-type ZSPI-TYP-CHAR50.

ZSPI-TKN-RETCODE          token-type ZSPI-TYP-RETCODE.

ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-LIST.
   ...
   ZSPI-TKN-ENDLIST       token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZSPI-TKN-COMMENT

is the standard SPI token that lets you include an 80-byte arbitrary comment in the command buffer. Although ORSERV ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-SERVER-BANNER

is a 50-character string that contains the standard version ID of the ORSERV server, the ORSERV release date, and the ORSERV compilation date:

ORSERV - T9074*vff* - *release-date* - *compilation-date*

where:

vff

is the ORSERV version ID. An example is D40.

release-date

is the ORSERV release date. An example is 01APR01.

compilation-date

is the ORSERV compilation date. An example is 01APR01.

ZSPI-TKN-RETCODE

is the standard SPI return token returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful, or an error number if an error or warning occurred.

Note. Table 4-4, Errors Returned by All ORSERV Commands, on page 4-21 shows the ZSPI-TKN-RETCODE values common to all ORSERV commands.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all ORSERV error numbers and their corresponding error lists, see Appendix C, ORSERV Error Messages.

ONLINERELOAD Command

Command

```
ZORS-CMD-ONLINERELOAD
```

Object Type

```
ZORS-OBJ-FILE
```

Tokens in the Command Buffer

```
ZORS-TKN-FILE           token-type ZSPI-TYP-FNAME.
ZORS-TKN-VOLUME         token-type ZSPI-TYP-CHAR8.

ZORS-MAP-PAR-ONLINERELOAD

def ZORS-DDL-PAR-ONLINERELOAD.
  02  ZNEW              type ZSPI-DDL-BOOLEAN.
  02  ZDUMP             type ZSPI-DDL-BOOLEAN.
  02  ZRATE             type ZSPI-DDL-INT.
  02  ZMIN-DSLACK       type ZSPI-DDL-INT.
  02  ZMAX-DSLACK       type ZSPI-DDL-INT.
  02  ZMIN-ISLACK       type ZSPI-DDL-INT.
  02  ZMAX-ISLACK       type ZSPI-DDL-INT.
  02  ZDEALLOCATE       type ZSPI-DDL-BOOLEAN
end.

ZSPI-TKN-MAXRESP        token-type ZSPI-TYP-INT.
ZSPI-TKN-RESPONSE-TYPE  token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE     token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT        token-type ZSPI-TYP-STRING.
```

Tokens in the Response Buffer

```
ZSPI-TKN-DATALIST       token-type ZSPI-TYP-LIST.
  ZORS-TKN-FILE         token-type ZSPI-TYP-FNAME.
  ZORS-TKN-VOLUME       token-type ZSPI-TYP-CHAR8.
  ZSPI-TKN-RETCODE      token-type ZSPI-TYP-RETCODE.

  ZSPI-TKN-ERRLIST      token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST    token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST      token-type ZSPI-TYP-SSCTL.
```

- It can be a single Enscribe file or an SQL object.
- It can be a primary or secondary partition of a partitioned file.
- It cannot have compressed keys in either data or index blocks.
- It can be audited; that is, it can be a file that is accessed under the control of the Compaq NonStop Transaction Management Facility (TMF).

- It can be unaudited. Reloading an unadited file might take longer than reloading an audited file.

Tokens in the Command Buffer

ZORS-TKN-FILE

is a required token that specifies a key-sequenced target file or SQL object for the reload operation. Only one of these tokens is allowed per command.

The target file must be a single file or SQL object in the Guardian internal file-name format. A file set or file-set list is not allowed.

ZORS-TKN-VOLUME

is the required volume name of the primary partition if ZORS-TKN-FILE specifies a secondary partition Enscribe file as the target file. If ZORS-TKN-FILE does not specify a secondary partition, ZORS-TKN-VOLUME is optional.

ZORS-MAP-PAR-ONLINERELOAD

is a required structured token that specifies options for the reload operation. Only one of these tokens is allowed per command. ORSERV only requires that the structure be nulled with SSNULL. The fields are:

ZNEW

is a Boolean field with these values:

ZSPI-VAL-TRUE

ORSERV starts a new reload operation for the file specified by ZORS-TKN-FILE.

ZSPI-VAL-FALSE

ORSERV restarts the reload operation for the file specified by ZORS-TKN-FILE at the point where the previous reload was suspended (if a previous reload was executed for the file). This is the default.

ZDUMP

is a Boolean field reserved for a future release.

ZRATE

specifies the percent of its execution time that ORSERV uses to perform the reload operation. ORSERV spends the remainder of its execution time in a delay mode. The range is 1 through 100, and 100 is the default.

When you restart a reload operation, specify a new value for ZRATE (or set ZRATE to -1) to use the value from the previous reload operation.

To minimize any performance degradation to your system caused by the reload operation, set the ZRATE field. For example, if ZRATE is 10, ORSERV uses 10 percent of its execution time to perform the reload and delays 90 percent of its time. This lets system resources (such as processor time and memory) be available for use by other applications.

ZMIN-DSLACK

specifies the minimum percentage of slack space to be left in data blocks. Allowable values are in the range 0 through 99. The default is 15. For the default value, set ZMIN-DSLACK to -1.

When you restart a reload operation, specify a new value for ZMIN-DSLACK (or set ZMIN-DSLACK to -1) to use the value from the previous reload operation.

ZMAX-DSLACK

specifies the maximum percentage of slack space to be left in data blocks. Allowable values are in the range 0 through 99, and the default is 15. For the default value, set ZMAX-DSLACK to -1.

When you restart a reload operation, specify a new value for ZMAX-DSLACK (or set ZMAX-DSLACK to -1) to use the value from the previous reload operation.

ZMIN-ISLACK

specifies the minimum percentage of slack space to be left in index blocks. Allowable values are in the range 0 through 99, and the default is 15. For the default value, set ZMIN-ISLACK to -1.

When you restart a reload operation, specify a new value for ZMIN-ISLACK (or set ZMIN-ISLACK to -1) to use the value from the previous reload operation.

ZMAX-ISLACK

specifies the maximum percentage of slack space to be left in index blocks. Allowable values are in the range 0 through 99, and the default is 15. For the default value, set ZMAX-ISLACK to -1.

When you restart a reload operation, specify a new value for ZMAX-ISLACK (or set ZMAX-ISLACK to -1) to use the value from the previous reload operation.

ZDEALLOCATE

specifies if the extents beyond the EOF should be kept or deallocated. This field is Boolean. Extents beyond the EOF are deallocated if this is true, but they are not if it is false.

ZSPI-TKN-MAXRESP

is the standard SPI token that specifies the number of responses a subsystem returns in the response buffer. Because ORSERV returns only one response record per buffer, ZSPI-TKN-MAXRESP can have these values:

Zero	ORSERV does not enclose the response in a data list.
Nonzero	ORSERV encloses the response in a data list.

`ZSPI-TKN-RESPONSE-TYPE and ZSPI-TKN-ALLOW-TYP`

are standard SPI response-control tokens. Although you can set these tokens, they have no effect on the execution of the command.

`ZSPI-TKN-COMMENT`

is the standard SPI token that lets you include an 80-byte arbitrary comment in the command buffer. Although ORSERV ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

`ZSPI-TKN-DATALIST`

is the standard SPI token that begins a data list. Because ORSERV returns only one response record per buffer, this response is enclosed in a data list if ZSPI-TKN-MAXRESP has a value other than zero. The data list ends with ZSPI-TKN-ENDLIST.

`ZORS-TKN-FILE`

is the name of the target file that is the object of the reload operation. This token is in the Guardian internal file-name format and is returned with every response.

`ZORS-TKN-VOLUME`

is the volume name of the primary partition if ZORS-TKN-FILE specified a secondary partition Enscribe file as the target file.

`ZSPI-TKN-RETCODE`

is the standard SPI return token returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful or an error number if an error or warning occurred.

When the ONLINERELOAD command is executed, ZSPI-TKN-RETCODE can also have the value ZORS-ERR-ORELOAD-INPROGRESS (15). This error indicates that an application specified the ONLINERELOAD command for a target file, but an online reload operation is already in progress for the file.

Note. Table 4-4, Errors Returned by All ORSERV Commands, on page 4-21 shows the ZSPI-TKN-RETCODE values common to all ORSERV commands.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all ORSERV error numbers and their corresponding error lists, see Appendix C, ORSERV Error Messages.

Considerations

When using the ONLINERELOAD command, consider the audit records, the ZZRELOAD file, and slack values.

Audit Records

The reload operation generates audit records describing the movement of data within the file. These records can be two or three times the length of the file (not counting free space).

ZZRELOAD File

ORSERV maintains information about the reload operation in the ZZRELOAD file, which is located on the same disk volume and subvolume as the target file. If ZZRELOAD does not exist when you execute the ONLINERELOAD command, ORSERV creates it.

Note. The structure of the ZZRELOAD file changed for the D20 release. This could cause a problem if ORSERV and the target file are not on the same node. Requests for STATUS, RELOAD, or SUSPEND operations do not execute properly unless you start ORSERV on the same node as the target file.

When ORSERV receives an ONLINERELOAD command, it uses the ZZRELOAD file:

1. Before it sends a response to the requester, ORSERV reads the ZZRELOAD file to determine if it contains information about a reload operation for the same target file as specified for the ONLINERELOAD command. ORSERV continues execution:
   - If ORSERV determines that a reload operation is already in progress for the target file, it returns the ZORS-ERR-ORELOAD-INPROGRESS error and terminates. The first reload operation continues for the target file.
   - If ORSERV determines that a previous reload operation for the same target file was stopped or suspended, it continues the reload operation depending on the value of the ZNEW Boolean field in the ZORS-MAP-PAR-ONLINERELOAD token. This value is one of:
     - ZSPI-VAL-TRUE—ORSERV starts the reload operation from the beginning even if a previous reload operation was suspended. ORSERV does not restart a reload operation that is already in progress.

- ZSPI-VAL-FALSE—ORSERV continues the reload operation from the point indicated in ZZRELOAD.

2. After ORSERV initiates the reload operation, it returns a response buffer to its requester. ORSERV then continues the reload operation and does not communicate with the requester. You can close ORSERV at this point.
3. ORSERV records any reload warnings or errors in the ZZRELOAD file.
4. After completing the reload operation, ORSERV stops itself.

After ORSERV completes the reload operation, the ZZRELOAD file remains on the same volume and subvolume as the target file. The data describing the reload operation is not overwritten unless a new reload operation is initiated for the same target file. This lets you use the STATUS command to get the status for the completed reload operation.

Slack Values

ORSERV sets the values as close as possible to the requested limits if it cannot meet the minimum and maximum slack space values.

Example

Figure 5-1 shows an example of a TAL integer procedure that sends an ONLINERELOAD command to ORSERV.

Figure 5-1. Example of the ONLINERELOAD Command (page 1 of 3)

```
! ---------------------------------------------------------
! ONLINERELOAD is a high-level integer procedure for the
! ORSERV ONLINERELOAD command.  It assumes that the ORSERV
! process was already created using PROCESS_LAUNCH_[NOWAIT]
! and is open.  This procedure returns 0 if successful,
! or an error number, if an error occurs.
! ---------------------------------------------------------
INT PROC ONLINERELOAD (target^file^name,
                       new,
                       rate,
                       min^dslack,
                       max^dslack,
                       min^islack,
                       max^islack,
                       deallocate)
                       VARIABLE;
! ---------------------------------------------------------
!  Parameter declarations.
! ---------------------------------------------------------
INT .target^file^name; ! Required input target file

! Optional input parameters:
INT new,        ! Start new reload operation
```

Figure 5-1. Example of the ONLINERELOAD Command (page 2 of 3)

```
    rate,       ! Percentage of execution time for reload
    min^dslack, ! Data slack minimum
    max^dslack, ! Data slack maximum
    min^islack, ! Index slack minimum
    max^islack, ! Index slack maximum
    deallocate; ! Deallocates extents beyond EOF
! ---------------------------------------------------------
BEGIN
! Global variables used are:
!   STRUCT .buffer (ZORS^DDL^MSG^BUFFER^DEF);
!   STRUCT .ORSERV^ssid (ZSPI^DDL^SSID^DEF);
!   INT     ORSERV^file^number,
!           error, spi^error;
!   LITERAL parameter^error = 99;
! ---------------------------------------------------------
! Local definitions.
! ---------------------------------------------------------
INT .onlinereload^par^def
                  [ 0 : (ZORS^MAP^PAR^ONLINERELOAD^WLN-1) ]
                                := ZORS^MAP^PAR^ONLINERELOAD;
STRUCT  .params (ZORS^DDL^PAR^ONLINERELOAD^DEF);
! ---------------------------------------------------------
! Check for the required parameters.
! ---------------------------------------------------------
IF NOT $PARAM (target^file^name) THEN
  RETURN parameter^error;
! ---------------------------------------------------------
! Format the command buffer for the ONLINERELOAD command.
! ---------------------------------------------------------
ORSERV^ssid ':=' [ ZSPI^VAL^TANDEM,
                   ZSPI^SSN^ZORS,
                   ZSPI^VAL^VERSION ];
CALL SSINIT (buffer,
             ZORS^VAL^BUFLEN,
             ORSERV^ssid,
             ZSPI^VAL^CMDHDR,
             ZORS^CMD^ONLINERELOAD,
             ZORS^OBJ^FILE);
! ---------------------------------------------------------
! Put the required file-name parameter into the buffer.
! ---------------------------------------------------------
  CALL SSPUTTKN (buffer,
                 ZORS^TKN^FILE,
                 target^file^name);
! ---------------------------------------------------------
! Put the optional parameters into the buffer.  First, call
! SSNULL to set the parameter structure to null values.
! ---------------------------------------------------------
IF $PARAM (new) OR $PARAM (rate) OR
   $PARAM (min^dslack) OR $PARAM (max^dslack) OR
   $PARAM (min^islack) OR $PARAM (max^islack) OR
   $PARAM (deallocate) THEN
```

Figure 5-1. Example of the ONLINERELOAD Command (page 3 of 3)

```
  BEGIN
  CALL SSNULL (onlinereload^par^def,
               params);
  IF $PARAM (new) THEN
     params.znew := new;
  IF $PARAM (rate) THEN
     params.zrate := rate;
  IF $PARAM (min^dslack) THEN
     params.zmin^dslack := min^dslack;
  IF $PARAM (max^dslack) THEN
     params.zmax^dslack := max^dslack;
  IF $PARAM (min^islack) THEN
     params.zmin^islack := min^islack;
  IF $PARAM (max^islack) THEN
     params.zmax^islack := max^islack;
  IF $PARAM (deallocate) THEN
     params.zdeallocate := deallocate;
CALL SSPUT (buffer,
            onlinereload^par^def,
            params);
 END;
! ---------------------------------------------------------
! Check for an SPI error.
! ---------------------------------------------------------
CALL SSGETTKN (buffer,
               ZSPI^TKN^LASTERR,
               spi^error);
IF spi^error THEN RETURN spi^error;
! ---------------------------------------------------------
! Call SEND^COMMAND to send the command buffer to ORSERV.
! ---------------------------------------------------------
error := send^command;
IF error THEN RETURN error;
! ---------------------------------------------------------
! Interpret the response buffer returned from ORSERV.
! ---------------------------------------------------------
error := SSGETTKN (buffer,
                   ZSPI^TKN^RETCODE,
                   return^error, 1);
IF error THEN CALL spi^proc^error^handler (error);
IF return^error THEN
  CALL retcode^error^handler (return^error);

RETURN 0; ! Successful return
END;      ! of ONLINERELOAD procedure.
```

STATUS Command

After receiving a STATUS command, ORSERV reads the ZZRELOAD file to determine the status of the reload operation for the specified target file. If ORSERV cannot read the ZZRELOAD file, it returns an error to the requester.

An ORSERV process accepts only one STATUS command. After returning the status in the response buffer, ORSERV stops itself.

Command

```
ZORS-CMD-STATUS
```

Object Type

```
ZORS-OBJ-FILE
```

Tokens in the Command Buffer

```
ZORS-TKN-FILE                token-type ZSPI-TYP-FNAME

ZSPI-TKN-MAXRESP             token-type ZSPI-TYP-INT.
ZSPI-TKN-RESPONSE-TYPE       token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE          token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT             token-type ZSPI-TYP-STRING.
```

Tokens in the Response Buffer

```
ZSPI-TKN-DATALIST            token-type ZSPI-TYP-LIST.
  ZORS-TKN-FILE              token-type ZSPI-TYP-FNAME.

  ZORS-MAP-STATUS-RESPONSE

  def ZORS-DDL-STATUS-RESPONSE.
    02  ZSTATUS-AVAILABLE        type ZSPI-DDL-BOOLEAN.
    02  ZACTIVE                  type ZSPI-DDL-BOOLEAN.
    02  ZINITIATION-TIMESTAMP    type ZSPI-DDL-INT4.
    02  ZSUSPENSION-TIMESTAMP    type ZSPI-DDL-INT4.
    02  ZABEND-TIMESTAMP         type ZSPI-DDL-INT4.
    02  ZRESUMPTION-TIMESTAMP    type ZSPI-DDL-INT4.
    02  ZCOMPLETION-TIMESTAMP    type ZSPI-DDL-INT4.
    02  ZPERCENT-DONE            type ZSPI-DDL-INT.
    02  ZERROR                   type ZSPI-DDL-INT2.
    02  ZERROR-FILE              type ZSPI-DDL-FNAME.
    02  ZMIN-DSLACK              type ZSPI-DDL-INT.
    02  ZMAX-DSLACK              type ZSPI-DDL-INT.
    02  ZMIN-ISLACK              type ZSPI-DDL-INT.

    02  ZMAX-ISLACK              type ZSPI-DDL-INT.
    02  ZRATE                    type ZSPI-DDL-INT.
    02  ZEYECATCHER              type ZSPI-DDL-INT
    02  ZSTATUS-VERSION          type ZSPI-DDL-INT
    02  ZDEALLOCATE              type ZSPI-DDL-BOOLEAN
  end.

  ZSPI-TKN-RETCODE          token-type ZSPI-TYP-RETCODE.
  ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST        token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZORS-TKN-FILE

is a required token that specifies the target file or SQL object for the STATUS command. Only one of these tokens is allowed per command.

The target file must be a single file or SQL object in the Guardian internal file-name format. A file set or file-set list is not allowed.

ZSPI-TKN-MAXRESP

is the standard SPI token that specifies the number of responses that a subsystem returns in the response buffer. Because ORSERV returns only one response per buffer, ZSPI-TKN-MAXRESP can have these values:

Zero	ORSERV does not enclose the response in a data list.
Nonzero	ORSERV encloses the response in a data list.

ZSPI-TKN-RESPONSE-TYPE and ZSPI-TKN-ALLOW-TYP

are standard SPI response-control tokens. Although you can set these tokens, they have no effect on the execution of the command.

ZSPI-TKN-COMMENT

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although ORSERV ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

ZSPI-TKN-DATALIST

is the standard SPI token that begins a data list. Because ORSERV returns only one response record per buffer, this response is enclosed in a data list if ZSPI-TKN-MAXRESP has a value other than zero. The data list ends with ZSPI-TKN-ENDLIST.

ZORS-TKN-FILE

specifies the target file that is the object of the STATUS command. This token is in the NonStop Kernel internal file-name format and is returned with every response.

ZORS-MAP-STATUS-RESPONSE

is a structured token that specifies the status of the reload operation. Only one of these tokens is returned in a response buffer. The fields are as follows:

ZSTATUS-AVAILABLE

is a Boolean field with these values:

ZSPI-VAL-TRUE

The status information is available, and ORSERV returns meaningful values for the remaining fields.

ZSPI-VAL-FALSE

The status information is not available, and ORSERV returns values set by the SSNULL procedure for the remaining fields.

ZACTIVE

is a Boolean field with these values:

ZSPI-VAL-TRUE

The reload operation is currently active.

ZSPI-VAL-FALSE

The reload operation is not active.

ZINITIATION-TIMESTAMP

is a Julian timestamp that specifies when the reload operation was first started.

ZSUSPENSION-TIMESTAMP

is a Julian timestamp that specifies when the reload operation was last suspended (if it was suspended).

ZABEND-TIMESTAMP

is a Julian timestamp that specifies when the reload operation abended (if it abended).

ZRESUMPTION-TIMESTAMP

is a Julian timestamp that specifies when the reload operation was last resumed after it was suspended (if it was suspended).

ZCOMPLETION-TIMESTAMP

is a Julian timestamp that specifies when the reload operation finished (if it completed).

ZPERCENT-DONE

specifies the percentage of the reload operation that has completed—if the reload is still executing.

ZERROR

specifies the last error that occurred for the reload operation.

ZERROR-FILE

is the file on which the last error occurred. This file is either the ZZRELOAD file or the target file specified by ZORS-TKN-FILE.

ZMIN-DSLACK

specifies the minimum percentage of slack space to be left in data blocks for the target file specified by ZORS-TKN-FILE.

ZMAX-DSLACK

specifies the maximum percentage of slack space to be left in data blocks for the target file specified by ZORS-TKN-FILE.

ZMIN-ISLACK

specifies the minimum percentage of slack space to be left in index blocks for the target file specified by ZORS-TKN-FILE.

ZMAX-ISLACK

specifies the maximum percentage of slack space to be left in index blocks for the target file specified by ZORS-TKN-FILE.

ZRATE

specifies the percent of its execution time that ORSERV uses for the reload operation. ORSERV spends the remainder of its execution time in a delay mode.

ZEYECATCHER

specifies an eyecatcher that is used internally.

ZSTATUS-VERSION

specifies the version of the ORSERV server that sent the status response.

ZDEALLOCATE

specifies the deallocate option of the RELOAD operation. Any extents beyond the EOF are deallocated after the RELOAD (if this is true). Any extents beyond the EOF are kept after the RELOAD (if this is false).

ZSPI-TKN-RETCODE

is the standard SPI return token that is returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful or an error number if an error or warning occurred.

Note. Table 4-4, Errors Returned by All ORSERV Commands, on page 4-21 shows the ZSPI-TKN-RETCODE values common to all ORSERV commands.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all ORSERV error numbers and their corresponding error lists, see Appendix C, ORSERV Error Messages.

Example

Figure 5-2 shows a TAL example of the STATUS command:

Figure 5-2. Example of the STATUS Command (page 1 of 2)

```
! ---------------------------------------------------------
! STATUS is a high-level integer procedure for the ORSERV
! STATUS command.
! ---------------------------------------------------------
INT PROC STATUS (target^file^name);
! ---------------------------------------------------------
INT .target^file^name; ! Required target file parameter
BEGIN
! ---------------------------------------------------------
! Global variables used are:
!   STRUCT .buffer (ZORS^DDL^MSG^BUFFER^DEF);
!   STRUCT .ORSERV^ssid (ZSPI^DDL^SSID^DEF);
!   INT     ORSERV^file^number,
!           error, return^error, spi^error;
! ---------------------------------------------------------!
Local definitions.
INT    .params^def [ 0:(ZORS^MAP^STATUS^RESPONSE^WLN-1) ];
STRUCT .params (ZORS^DDL^STATUS^RESPONSE^DEF);
! ---------------------------------------------------------
! Format the buffer for the STATUS command.
! ---------------------------------------------------------

ORSERV^ssid ':=' [ ZSPI^VAL^TANDEM,
                   ZSPI^SSN^ZORS,
                   ZSPI^VAL^VERSION ];
error := SSINIT (buffer,
                 ZORS^VAL^BUFLEN,
```

Figure 5-2. Example of the STATUS Command (page 2 of 2)

```
                ORSERV^ssid,
                ZSPI^VAL^CMDHDR,
                ZORS^CMD^STATUS,
                ZORS^OBJ^FILE);
IF error THEN CALL spi^proc^error^handler (error);

! ----------------------------------------------------------
! Put the file name parameter into the buffer.
! ----------------------------------------------------------
error := SSPUTTKN (buffer,
                   ZORS^TKN^FILE,
                   target^file^name);
IF error THEN CALL spi^proc^error^handler (error);
! ----------------------------------------------------------
! Get the last error from the buffer.
! ----------------------------------------------------------
error := SSGETTKN  (buffer,
                    ZSPI^TKN^LASTERR,
                    spi^error );
IF error THEN CALL spi^proc^error^handler (error);
IF spi^error THEN CALL spi^proc^error^handler (spi^error);
! ----------------------------------------------------------
! Send the command buffer to ORSERV.
! ----------------------------------------------------------
error := send^command;

! ----------------------------------------------------------
! Interpret the response buffer; first check for errors
! ----------------------------------------------------------
error := SSGETTKN (buffer,
                   ZSPI^TKN^RETCODE,
                   return^error, 1);
IF error THEN CALL spi^proc^error^handler (error);
IF return^error <> ZORS^ERR^OK THEN
  CALL retcode^error^handler (return^error);
! ----------------------------------------------------------
! Get the status token from the return buffer.
! ----------------------------------------------------------

error := SSGET (buffer,
                params^def,
                params, 1);
IF error THEN CALL spi^proc^error^handler (error);

! Process the status information.

RETURN 0; ! Successful return.
END;      ! of STATUS procedure.
```

SUSPEND Command

ORSERV reads the ZZRELOAD file to determine the process ID of the ORSERV process performing the reload operation, then stops it. ORSERV records the timestamp when the process is stopped in the ZZRELOAD file.

An ORSERV process accepts only one SUSPEND command. After stopping the process performing the reload operation (and returning the response buffer), ORSERV stops itself.

Command

```
ZORS-CMD-SUSPEND
```

Object Type

```
ZORS-OBJ-FILE
```

Tokens in the Command Buffer

```
ZORS-TKN-FILE              token-type ZSPI-TYP-FNAME
ZSPI-TKN-MAXRESP           token-type ZSPI-TYP-INT.
ZSPI-TKN-RESPONSE-TYPE     token-type ZSPI-TYP-ENUM.
ZSPI-TKN-ALLOW-TYPE        token-type ZSPI-TYP-ENUM.
ZSPI-TKN-COMMENT           token-type ZSPI-TYP-STRING.
```

Tokens in the Response Buffer

```
ZSPI-TKN-DATALIST          token-type ZSPI-TYP-LIST.
  ZORS-TKN-FILE            token-type ZSPI-TYP-FNAME.
  ZSPI-TKN-RETCODE         token-type ZSPI-TYP-RETCODE.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    ...
    ZSPI-TKN-ENDLIST       token-type ZSPI-TYP-SSCTL.
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
```

Tokens in the Command Buffer

ZORS-TKN-FILE

is a required token that specifies the target file or SQL object for the SUSPEND command. Only one of these tokens is allowed per command.

The target file must be a single file or SQL object in the Guardian internal file-name format. A file set or file-set list is not allowed.

ZSPI-TKN-MAXRESP

is the standard SPI token that specifies the number of responses that a subsystem returns in the response buffer. Because ORSERV returns only one response per buffer, ZSPI-TKN-MAXRESP can have these values:

Zero	ORSERV does not enclose the response in a data list.
Nonzero	ORSERV encloses the response in a data list.

`ZSPI-TKN-RESPONSE-TYPE and ZSPI-TKN-ALLOW-TYPE`

are standard SPI response-control tokens. Although you can set these tokens, they have no effect on the execution of the command.

`ZSPI-TKN-COMMENT`

is the standard SPI token that allows you to include an 80-byte arbitrary comment in the command buffer. Although ORSERV ignores ZSPI-TKN-COMMENT in the command buffer, the token can be useful when you are debugging an application.

Tokens in the Response Buffer

`ZSPI-TKN-DATALIST`

is the standard SPI token that begins a data list. Because ORSERV returns only one response record per buffer, this response is enclosed in a data list if ZSPI-TKN-MAXRESP has a value other than zero. The data list ends with ZSPI-TKN-ENDLIST.

`ZORS-TKN-FILE`

specifies the target file that is the object of the SUSPEND command. Only one of these tokens is returned in a response buffer. This token is in the Guardian internal file-name format and is returned with every response.

`ZSPI-TKN-RETCODE`

is the standard SPI return token that is returned in the response buffer by all Compaq subsystems. ZSPI-TKN-RETCODE contains zero if the command was successful or an error number if an error or warning occurred.

When the SUSPEND command is executed, ZSPI-TKN-RETCODE can also have these values:

- ZORS-ERR-NO-ORELOAD (16)

  An application specified the SUSPEND command for a target file, but an online reload operation had not been previously initiated for the file.

- ZORS-ERR-CANT-SUSPEND (17)

  An application specified the SUSPEND command for a target file, but the online reload operation cannot be suspended because an error occurred on:

  - The STOP procedure call when ORSERV tried to stop the process performing the reload operation.

- A procedure call (such as KEYPOSITION) when ORSERV tried to access the ZZRELOAD file.

Note. Table 4-4, Errors Returned by All ORSERV Commands, on page 4-21 shows the ZSPI-TKN-RETCODE values common to all ORSERV commands.

ZSPI-TKN-ERRLIST

is the standard SPI token that begins an error list. The error list ends with ZSPI-TKN-ENDLIST.

Note. For more information about all ORSERV error numbers and their corresponding error lists, see Appendix C, ORSERV Error Messages.

Example

Figure 5-3 shows a TAL example of the SUSPEND command:

Figure 5-3. Example of the SUSPEND Command (page 1 of 2)

```
! ----------------------------------------------------------
! SUSPEND^ORSERV is a high-level integer procedure for the
! SUSPEND command.  It assumes that the ORSERV process
! has already been created using PROCESS_LAUNCH_ [NOWAIT] and is
! open.  This procedure returns 0 IF successful,
! or an error number, if an error occurs.
! ----------------------------------------------------------
INT PROC SUSPEND^ORSERV (target^file^name) VARIABLE;
! ----------------------------------------------------------
INT .target^file^name; ! Required file name parameter
BEGIN
! ----------------------------------------------------------
! Global variables used are:
!   STRUCT .buffer (ZORS^DDL^MSG^BUFFER^DEF);
!   INT     ORSERV^file^number,
!           error, spi^error;
!   LITERAL parameter^error = 99;
! ----------------------------------------------------------
! Check for the required target^file^name parameter.
! ----------------------------------------------------------
IF NOT $PARAM (target^file^name) THEN
  RETURN parameter^error;
! ----------------------------------------------------------
! Format the command buffer for the SUSPEND command.
! ----------------------------------------------------------
ORSERV^ssid ':=' [ ZSPI^VAL^TANDEM,
                   ZSPI^SSN^ZORS,
                   ZSPI^VAL^VERSION ];
```

Figure 5-3. Example of the SUSPEND Command (page 2 of 2)

```
error := SSINIT (buffer,
                 ZORS^VAL^BUFLEN,
                 ORSERV^ssid,
                 ZSPI^VAL^CMDHDR,
                 ZORS^CMD^SUSPEND,
                 ZORS^OBJ^FILE);
IF error THEN CALL spi^proc^error^handler (error);

! ---------------------------------------------------------
! Put the required file-name parameter into the buffer.
! ---------------------------------------------------------
error := SSPUTTKN (buffer,
                   ZORS^TKN^FILE,
                   target^file^name);
IF error THEN CALL spi^proc^error^handler (error);
! ---------------------------------------------------------
! Check for an SPI error.
! ---------------------------------------------------------
error := SSGETTKN (buffer,
                   ZSPI^TKN^LASTERR,
                   spi^error);
IF error THEN CALL spi^proc^error^handler (error);
IF spi^error THEN RETURN spi^error;
! ---------------------------------------------------------
! Call SEND^COMMAND to send the command buffer to ORSERV.
! ---------------------------------------------------------
error := send^command;
IF error THEN RETURN error;
! ---------------------------------------------------------
! Interpret the response buffer returned from ORSERV.
! ---------------------------------------------------------
error := SSGETTKN (buffer,
                   ZSPI^TKN^RETCODE,
                   return^error, 1);
IF error THEN CALL spi^proc^error^handler (error);
IF return^error <> ZORS^ERR^OK THEN
  CALL retcode^error^handler (return^error);

RETURN 0; ! Successful return
END;      ! of SUSPEND^ORSERV procedure.
```

A

Management Application Example

This listing output is an example of using the COBOL85 management application to communicate with FUP and execute the programmatic DUPLICATE command.

Figure A-1. Management Application Example (page 1 of 8)

```
*=========================================================
* FUP DUPLICATE Program
*=========================================================
?SYMBOLS, INSPECT
?SAVE STARTUP
?LIBRARY $SYSTEM.SYSTEM.COBOLLIB

 IDENTIFICATION DIVISION.
 PROGRAM-ID. FUP-COPY.

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. TANDEM-16.
 OBJECT-COMPUTER. TANDEM-16.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT FUP-FILE ASSIGN TO #DYNAMIC.

 DATA DIVISION.
 FILE SECTION.
 FD  FUP-FILE
     LABEL RECORDS ARE OMITTED.
 COPY ZFUP-DDL-MSG-BUFFER OF "$SYSTEM.ZSPIDEF.ZFUPCOB".
 WORKING-STORAGE SECTION.
 01 ERR                     NATIVE-2.
 01 ERR-1                   NATIVE-2.
 01 ERR-2                   NATIVE-2.
 01 IN-PORTION              PIC X(2) VALUE "IN".
 01 VOLUME-PORTION          PIC X(6) VALUE "VOLUME".
 01 CP-LIST                 PIC 9(9) COMP VALUE 0.
 01 RCV-NAME                PIC X(8) VALUE "$RECEIVE".
 01 FUP-NAME                PIC X(30)
                            VALUE "$SYSTEM.SYSTEM.FUP".
 01 FUP-PROCESS-NAME        PIC X(6).
 01 FUP-SPI-NAME            PIC X(12).
 01 EXTERNAL-DEFAULT-VOL    PIC X(30).
 01 TEMP                    PIC X(30).
 01 JUNK                    PIC X(16) VALUE SPACES.
 01 LEN                     NATIVE-2.
 01 DEFAULT-VOL             PIC X(24).
 01 USER-TYPED-FILENAME     PIC X(40).
 01 Y-OR-N                  PIC X.
 01 DEST-OPTION             PIC X.
 01 COBOL-VAL-TRUE          PIC XX VALUE HIGH-VALUES.
```

Figure A-1. Management Application Example (page 2 of 8)

```
 01 COBOL-VAL-FALSE        PIC XX VALUE LOW-VALUES.
 01 RETCODE                NATIVE-2.
 01 EXTERNAL-NAME          PIC X(35).
 01 IN-EXTERNAL-NAME       PIC X(35).
 01 OUT-EXTERNAL-NAME      PIC X(35).

*============================================================
*  Copy SPI definitions from ZSPICOB, ZFUPCOB and ZFILCOB.
*============================================================
 COPY ZSPI-DDL-FNAME     OF "$SYSTEM.ZSPIDEF.ZSPICOB"
   REPLACING ZSPI-DDL-FNAME BY SOURCE-FILE.
 COPY ZSPI-DDL-FNAME     OF "$SYSTEM.ZSPIDEF.ZSPICOB"
   REPLACING ZSPI-DDL-FNAME BY DEST-FILE.
 COPY ZSPI-DDL-FNAME     OF "$SYSTEM.ZSPIDEF.ZSPICOB"
   REPLACING ZSPI-DDL-FNAME BY ERROR-FILE.
 COPY ZFUP-DDL-PAR-DUP   OF "$SYSTEM.ZSPIDEF.ZFUPCOB".
 COPY ZSPI-DDL-ERROR     OF "$SYSTEM.ZSPIDEF.ZSPICOB".
 COPY CONSTANTS          OF "$SYSTEM.ZSPIDEF.ZFUPCOB".
 COPY CONSTANTS          OF "$SYSTEM.ZSPIDEF.ZSPICOB".
 COPY CONSTANTS          OF "$SYSTEM.ZSPIDEF.ZFILCOB".
*============================================================
*  Procedure Division
*============================================================
 PROCEDURE DIVISION.
 P000.
*============================================================
*  Set IN file of saved startup message to "$RECEIVE"
*  (for when it is sent to FUP by CREATEPROCESS).
  ENTER "PUTSTARTUPTEXT" USING IN-PORTION
                               RCV-NAME
                               CP-LIST
                         GIVING ERR.
  IF ERR < 0  DISPLAY "PUTSTARTUPTEXT error: " ERR
              DISPLAY "Operation terminated."
              STOP RUN.
*============================================================
*  Get a process name to use when starting FUP.
  ENTER TAL "CREATEPROCESSNAME" USING FUP-PROCESS-NAME.
*============================================================
*  Start the FUP process.
  ENTER "CREATEPROCESS" USING FUP-NAME
                              FUP-PROCESS-NAME
                              1
                        GIVING ERR.
  IF ERR NOT = 0
     IF ERR < 256  DISPLAY "CREATEPROCESS error: " ERR
                   DISPLAY "Operation terminated."
                   STOP RUN
     ELSE DIVIDE ERR BY 256 GIVING ERR-1 REMAINDER ERR-2
          DISPLAY "NEWPROCESS error (" ERR-1 "," ERR-2 ")"
          DISPLAY "Operation terminated."
          STOP RUN.
```

Figure A-1. Management Application Example (page 3 of 8)

```
*============================================================
*  Form the name to open FUP for SPI commands and assign
*  it to the #DYNAMIC file.
  STRING FUP-PROCESS-NAME  ".#ZSPI" DELIMITED BY " "
    INTO FUP-SPI-NAME.
  ENTER "COBOLASSIGN" USING FUP-FILE FUP-SPI-NAME
                      GIVING ERR.
  IF ERR NOT = 0   DISPLAY "COBOLASSIGN error: " ERR
                   DISPLAY "Operation terminated."
                   STOP RUN.
*============================================================
*  Open the FUP server.
  OPEN I-O FUP-FILE.
*============================================================
*  Get the default volume and subvolume from the startup
*  message.  Convert them into internal format by
*  appending a file name and then calling FNAMEEXPAND.
  ENTER "GETSTARTUPTEXT" USING VOLUME-PORTION
                               EXTERNAL-DEFAULT-VOL
                         GIVING ERR.
  IF ERR < 0  DISPLAY "GETSTARTUPTEXT error: " ERR
              DISPLAY "Operation terminated."
              STOP RUN.
  STRING EXTERNAL-DEFAULT-VOL ".X" DELIMITED BY " "
    INTO TEMP.
  ENTER TAL "FNAMEEXPAND" USING TEMP
                                DEFAULT-VOL
                                JUNK
                          GIVING LEN.
  IF LEN = 0 DISPLAY "FNAMEEXPAND error occurred."
             DISPLAY "Operation terminated."
             STOP RUN.
*============================================================
*  Ask the user for the source file name, or Q[UIT] to
*  stop.  Convert the name to the internal format.
 P100.
  DISPLAY "FUP DUPLICATE Program".
  DISPLAY "Enter the source file name (or QUIT to stop):".
  ACCEPT USER-TYPED-FILENAME.
  enter tal "debug".
  ENTER TAL "SHIFTSTRING"
    USING USER-TYPED-FILENAME, 35, 0.
  EVALUATE USER-TYPED-FILENAME
    WHEN "QUIT" DISPLAY "Good-bye."
                CLOSE FUP-FILE
                STOP RUN
    WHEN "Q"    DISPLAY "Good-bye."
                CLOSE FUP-FILE
                STOP RUN
    END-EVALUATE.
```

Figure A-1. Management Application Example (page 4 of 8)

```
  ENTER TAL "FNAMEEXPAND" USING USER-TYPED-FILENAME
                               SOURCE-FILE
                               DEFAULT-VOL
                            GIVING LEN.
  IF LEN = 0 DISPLAY "Invalid file name.  Try again."
             GO TO P100.
*============================================================
*  Ask the user for the destination file name and convert
*  the name to internal format.
 P200.
  DISPLAY "Enter the destination file name:".
  ACCEPT USER-TYPED-FILENAME.

  ENTER TAL "SHIFTSTRING" USING USER-TYPED-FILENAME, 35, 0.

  ENTER TAL "FNAMEEXPAND" USING USER-TYPED-FILENAME
                               DEST-FILE
                               DEFAULT-VOL
                            GIVING LEN.
  IF LEN = 0 DISPLAY "Invalid file name.  Try again."
             GO TO P200.

*============================================================
*  Get the FUP options for the DUPLICATE command.
*  First, call the SSNULL procedure to initialize
*  the options structure to null values.
  ENTER TAL "SSNULL"
    USING ZFUP-MAP-PAR-DUP, ZFUP-DDL-PAR-DUP
    GIVING ERR.
  IF ERR NOT = 0 DISPLAY "SSNULL error: " ERR
                 DISPLAY "Operation terminated."
                 GO TO P700.
 P300.
  DISPLAY "Do you want to preserve the file timestamp?".
  ACCEPT Y-OR-N.
  ENTER TAL "SHIFTSTRING" USING Y-OR-N, 1, 0.
  EVALUATE Y-OR-N
     WHEN "Y"  MOVE COBOL-VAL-TRUE
       TO ZPRESERVE-TIMESTAMP OF ZFUP-DDL-PAR-DUP
     WHEN "N"  MOVE COBOL-VAL-FALSE
       TO ZPRESERVE-TIMESTAMP OF ZFUP-DDL-PAR-DUP
     WHEN " "  CONTINUE
     WHEN OTHER DISPLAY "Enter Y or N (or blank for N)."
                GO TO P300
     END-EVALUATE.
```

Figure A-1. Management Application Example (page 5 of 8)

```
 P400.
  DISPLAY "Do you want to preserve the file owner?".
  ACCEPT Y-OR-N.
  ENTER TAL "SHIFTSTRING" USING Y-OR-N, 1, 0.
  EVALUATE Y-OR-N
     WHEN "Y"  MOVE COBOL-VAL-TRUE
       TO ZPRESERVE-OWNER OF ZFUP-DDL-PAR-DUP
     WHEN "N"  MOVE COBOL-VAL-FALSE
       TO ZPRESERVE-OWNER OF ZFUP-DDL-PAR-DUP
     WHEN " "  CONTINUE
     WHEN OTHER DISPLAY "Enter Y or N (or blank for N)."
                GO TO P400
     END-EVALUATE.
 P500.
  DISPLAY "Do you want to preserve the file security?".
  ACCEPT Y-OR-N.
  ENTER TAL "SHIFTSTRING" USING Y-OR-N, 1, 0.
  EVALUATE Y-OR-N
     WHEN "Y"  MOVE COBOL-VAL-TRUE
       TO ZPRESERVE-SECURITY OF ZFUP-DDL-PAR-DUP
     WHEN "N"  MOVE COBOL-VAL-FALSE
       TO ZPRESERVE-SECURITY OF ZFUP-DDL-PAR-DUP
     WHEN " "  CONTINUE
     WHEN OTHER DISPLAY "Enter Y or N (or blank for N)."
                GO TO P500
     END-EVALUATE.
P600.
  DISPLAY "Destination file option:"
  DISPLAY "N=New, P=Purge, O=Old, or K=Keep?".
  DISPLAY "Enter N, P, O, or K (or blank for New)."
  ACCEPT DEST-OPTION.
  ENTER TAL "SHIFTSTRING" USING DEST-OPTION, 1, 0.
  EVALUATE DEST-OPTION

     WHEN "N"   MOVE ZFUP-VAL-NEW
       TO ZDEST-OPTION OF ZFUP-DDL-PAR-DUP
     WHEN "P"   MOVE ZFUP-VAL-PURGE
       TO ZDEST-OPTION OF ZFUP-DDL-PAR-DUP
     WHEN "O"   MOVE ZFUP-VAL-OLD
       TO ZDEST-OPTION OF ZFUP-DDL-PAR-DUP
     WHEN "K"   MOVE ZFUP-VAL-KEEP
        TO ZDEST-OPTION OF ZFUP-DDL-PAR-DUP
     WHEN “ “ CONTINUE
     WHEN OTHER GO TO P600
    END-EVALUATE.
```

Figure A-1. Management Application Example (page 6 of 8)

```
*=========================================================
*  Build the command buffer.
  ENTER TAL "SSINIT" USING ZFUP-DDL-MSG-BUFFER
                           ZFUP-VAL-BUFLEN
                           ZFUP-VAL-SSID
                           0
                           ZFUP-CMD-DUPLICATE
                           ZFUP-OBJ-FILE
                     GIVING ERR.
  IF ERR NOT = 0 DISPLAY "SSINIT error: " ERR
                 DISPLAY "Operation terminated."
                 GO TO P700.

  ENTER TAL "SSPUT" USING ZFUP-DDL-MSG-BUFFER
                          ZFUP-TKN-SOURCE-FILE
                          SOURCE-FILE
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSPUT error on ZFUP-TKN-SOURCE-FILE: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.

  ENTER TAL "SSPUT" USING ZFUP-DDL-MSG-BUFFER
                          ZFUP-TKN-DEST-FILE
                          DEST-FILE
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSPUT error on ZFUP-TKN-DEST-FILE: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.

  ENTER TAL "SSPUT" USING ZFUP-DDL-MSG-BUFFER
                          ZFUP-MAP-PAR-DUP
                          ZFUP-DDL-PAR-DUP
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSPUT error on ZFUP-MAP-PAR-DUP: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.
*=========================================================
*   Send the command buffer to FUP.
  READ FUP-FILE WITH PROMPT ZFUP-DDL-MSG-BUFFER.
*=========================================================
*  The FUP response is in ZFUP-DDL-MSG-BUFFER. Call the
*  SSPUT procedure with ZSPI-TKN-RESET-BUFFER.
  ENTER TAL “SSPUT” USING ZFUP-DDL-MSG-BUFFER
                          ZSPI-TKN-RESET-BUFFER
                          ZFUP-VAL-BUFLEN
                    GIVING ERR.
  IF ERR NOT = 0 DISPLAY “Reset buffer error: “ ERR
                 DISPLAY “Operation terminated.”
                 GO TO P700.
```

Figure A-1. Management Application Example (page 7 of 8)

```
*=========================================================
*  Get the return code token.  Specify an index of 1 in
*  the SSGET call so that the order of tokens in the
*  buffer doesn't matter.
  ENTER TAL "SSGET" USING ZFUP-DDL-MSG-BUFFER
                          ZSPI-TKN-RETCODE
                          RETCODE
                          1
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSGET error on ZSPI-TKN-RETCODE: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.
  IF RETCODE NOT = 0
    EVALUATE RETCODE
      WHEN ZFUP-ERR-FILESYS  PERFORM DISPLAY-FILESYS-ERROR
      WHEN OTHER
        DISPLAY "DUP failed; ZSPI-TKN-RETCODE: " RETCODE
      END-EVALUATE
    DISPLAY "Operation terminated."
    GO TO P700.
*=========================================================
 *  Display the successful DUP message.
  DISPLAY "DUP was successful!".
  DISPLAY " ".
  MOVE SPACES TO IN-EXTERNAL-NAME.
  ENTER TAL "FNAMECOLLAPSE" USING SOURCE-FILE
                                  IN-EXTERNAL-NAME.
  MOVE SPACES TO OUT-EXTERNAL-NAME.
  ENTER TAL "FNAMECOLLAPSE" USING DEST-FILE
                                  OUT-EXTERNAL-NAME.
  DISPLAY
    IN-EXTERNAL-NAME " duplicated to " OUT-EXTERNAL-NAME.
*=========================================================
*  Determine if the user wants to dup another file.
 P700.
  DISPLAY " ".
  DISPLAY "Do you want to DUP another file (Y or N)?".
  ACCEPT Y-OR-N.
  ENTER TAL "SHIFTSTRING" USING Y-OR-N, 1, 0.
  EVALUATE Y-OR-N
     WHEN "Y" GO TO P100
     WHEN "N" DISPLAY "Good-bye."
              CLOSE FUP-FILE
              STOP RUN
     WHEN OTHER DISPLAY "Please enter Y or N."
                GO TO P700
     END-EVALUATE.
*=========================================================
*  DISPLAY-FILESYS-ERROR routine - Enters the FUP error
*  list and then enters the nested file-system error list.
*  From the file-system error list, gets the ZSPI-TKN-ERROR
```

Figure A-1. Management Application Example (page 8 of 8)

```
*  token, which contains the file-system error number and
*  the ZFIL-TKN-FILENAME token, which contains the name of
*  the file that FUP was processing when the error occurred.
 DISPLAY-FILESYS-ERROR.
  ENTER TAL "SSGET" USING  ZFUP-DDL-MSG-BUFFER
                           ZSPI-TKN-ERRLIST
                           OMITTED
                           1
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSGET error on the FUP error list: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.
  ENTER TAL "SSGET" USING ZFUP-DDL-MSG-BUFFER
                          ZSPI-TKN-ERRLIST
                          OMITTED
                          1
                          OMITTED
                          ZFIL-VAL-SSID
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSGET error on file-system error list: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.
  ENTER TAL "SSGET" USING ZFUP-DDL-MSG-BUFFER
                          ZSPI-TKN-ERROR
                          ZSPI-DDL-ERROR
                          1
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSGET error for ZSPI-TKN-ERROR: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.
  ENTER TAL "SSGET" USING ZFUP-DDL-MSG-BUFFER
                          ZFIL-TKN-FILENAME
       ERROR-FILE
                          1
                    GIVING ERR.
  IF ERR NOT = 0
    DISPLAY "SSGET error for ZFIL-TKN-FILENAME: " ERR
    DISPLAY "Operation terminated."
    GO TO P700.

  MOVE SPACES TO EXTERNAL-NAME.
  ENTER TAL "FNAMECOLLAPSE" USING ERROR-FILE
                                  EXTERNAL-NAME.
  DISPLAY "File-system error: " Z-ERROR OF ZSPI-DDL-ERROR
  DISPLAY "File name:         " EXTERNAL-NAME.

*============================================================
* End of FUP DUPLICATE Program
*============================================================
```

B FUP Error Messages

The FUP process returns an error list in the response buffer when it detects an error in the command syntax or during execution of a command. The error lists are organized by error number (ZFUP-ERR-*value*) in ascending order. Each error-list description contains:

- A header displaying the numeric and symbolic values for the error
- A paragraph identifying the cause of the error
- A box listing the simple and structured tokens that can appear in the error list

  The notation used in the box for simple tokens is a shorthand version of the essential information given in the DDL TOKEN-CODE statement.

  Each error list begins with the ZSPI-TKN-ERRLIST token and ends with the ZSPI-TKN-ENDLIST token.

  Any nested error lists from other subsystems or software components (including the FASTSORT utility) also begin and end with these tokens.

  The remaining tokens are not in any particular order of occurrence.
- A description of each token in the error list (excluding ZSPI-TKN-ERRLIST and ZSPI-TKN-ENDLIST)
- A recommended course of action to correct or alleviate the error

The FUP process sets the return token ZSPI-TKN-RETCODE by the error number to identify the error list. If a command finishes without an error, FUP sets ZSPI-TKN-RETCODE to ZFUP-ERR-OK (0).

Although ZSPI-TKN-RETCODE is zero, the response buffer can still contain an error list. You need to investigate this warning. It signals a condition that does not prevent FUP from executing commands but could cause other problems.

To determine the problem, enter the error list and check the value of the ZSPI-TKN-ERROR token.

Note. For more information about FUP errors, see Section 2, FUP Programmatic Interface.

1: ZFUP-ERR-INV-COMMAND

An application sent an invalid programmatic command to FUP.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INV-COMMAND (1). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZCOMMAND

is the invalid command number.

ZNAME and ZOBJECT

are not used and have null values.

Recommended Action

Correct the invalid command in the SSINIT procedure call and retry the command.

2: ZFUP-ERR-INV-OBJECT

An application specified an invalid object type for a FUP programmatic command.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INV-OBJECT (2). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZOBJECT

is the invalid object type.

ZNAME and ZCOMMAND

are not used and have null values.

Recommended Action

The valid object type for the DUPLICATE, CHECKSUM, LOAD, LOADALTFILE, and RESTART commands is ZFUP-OBJ-FILE. The valid object type for the GETVERSION command is ZFUP-OBJ-NULL. Correct the invalid object type and retry the command.

3: ZFUP-ERR-INVALID-TOKEN

An application specified an invalid token in the command buffer.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE             type ZSPI-DDL-TOKENCODE.
    Z-INDEX                 type ZSPI-DDL-UINT.
    Z-OFFSET                type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INVALID-TOKEN (3). This token is always in the error list.

ZSPI-TKN-PARM-ERR

is the standard SPI parameter error token. It contains these fields:

Z-TOKENCODE

is the token code that caused the error.

Z-INDEX and Z-OFFSET

are not used and have null values.

Recommended Action

Determine the invalid token code from Z-TOKENCODE in the error list. Omit this token or substitute the correct token and retry the command.

4: ZFUP-ERR-MISS-TOKEN

An application omitted a required token from the command buffer.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE             type ZSPI-DDL-TOKENCODE.
    Z-INDEX                 type ZSPI-DDL-UINT.
    Z-OFFSET                type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-MISS-TOKEN (4). This token is always in the error list.

`ZSPI-TKN-PARM-ERR`

is the standard SPI parameter error token. It contains these fields:

`Z-TOKENCODE`

is the missing token code that caused the error.

`Z-INDEX` and *`Z-OFFSET`*

are not used and have null values.

Recommended Action

Determine the missing token from Z-TOKENCODE in the error list. Add this token and retry the command.

5: ZFUP-ERR-MISS-FIELD

An application failed to set a required field of a structure to a specific value, and the field has a null value.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-MISS-FIELD (5). This token is always in the error list.

ZSPI-TKN-PARM-ERR

is the standard SPI parameter error token. It contains these fields:

Z-TOKENCODE

is the token code that caused the error.

Z-INDEX

is the occurrence number of the token that caused the error.

Z-OFFSET

is the byte offset of the field that caused the error.

Recommended Action

Determine the field that has a null value from the Z-TOKENCODE and Z-OFFSET tokens in the error list. Supply this field with a value and retry the command.

6: ZFUP-ERR-EXTRA-TOKEN

An application specified an extra token in the command buffer.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-EXTRA-TOKEN (6). This token is always in the error list.

ZSPI-TKN-PARM-ERR

is the standard SPI parameter error token. It contains these fields:

Z-TOKENCODE

is the extra token code that caused the error.

Z-INDEX

is the occurrence number of the token that caused the error.

Z-OFFSET

is not used and has a null value.

Recommended Action

Determine the extra token from Z-TOKENCODE in the error list. Omit this token from the command buffer and retry the command.

7: ZFUP-ERR-INV-VALUE

An application set a token or field within a token to an invalid value.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INV-VALUE (7). This token is always in the error list.

`ZSPI-TKN-PARM-ERR`

is the standard SPI parameter error token. It contains these fields:

`Z-TOKENCODE`

is the token code of the token that caused the error.

`Z-INDEX`

is the occurrence number of the token that caused the error.

`Z-OFFSET`

is the byte offset of the field that caused the error. This token is null if the field is not part of a structured token.

Recommended Action

Determine the token and field (if applicable) that has the invalid value from the Z-TOKENCODE and Z-OFFSET tokens in the error list. Correct this token or field and retry the command.

8: ZFUP-ERR-INV-CONTEXT

An application specified an invalid context token in the command buffer.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INV-CONTEXT (8). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is not used and has a null value.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Try to determine why the context token is invalid. For example, determine if the context token was inadvertently overwritten in the buffer.

9: ZFUP-ERR-INV-TEMPLATE

An application specified an invalid template name for a file set.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INV-TEMPLATE (9). This token is always in the error list.

`ZSPI-TKN-PARM-ERR`

is the standard SPI parameter error token. It contains these fields:

`Z-TOKENCODE`

is the invalid token code that caused the error.

`Z-INDEX`

is the occurrence number of the token that caused the error.

`Z-OFFSET`

is not used.

Recommended Action

You can specify a file-set template for the CHECKSUM and DUPLICATE commands. Correct the template and retry the command.

10: ZFUP-ERR-LONG-COMMAND

An application specified a command that was too long for the buffer.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-LONG-COMMAND (10). This token is always in the error list.

Recommended Action

The recommended buffer length for all FUP commands is ZFUP-VAL-BUFLEN. If your buffer is shorter, reallocate it and retry the command.

11: ZFUP-ERR-WRONG-SSID

An application specified an incorrect FUP subsystem ID (SSID) for a command.

```
ZSPI-TKN-ERRLIST       token-type ZSPI-TYP-LIST.
ZSPI-TKN-ERROR        token-type ZSPI-TYP-ERROR.
ZSPI-TKN-ENDLIST       token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-WRONG-SSID (11). This token is always in the error list.

Recommended Action

The correct FUP subsystem ID is ZFUP-VAL-SSID. Correct the SSINIT procedure call and retry the command.

12: ZFUP-ERR-WRONG-SERVER

An application specified a command that requires a newer version of the FUP server.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-WRONG-SERVER (12). This token is always in the error list.

Recommended Action

You must use a newer version of FUP for this command. Determine the correct server version, start that version, and retry the command.

13: ZFUP-ERR-EMPTY-RESP

This error number indicates an empty response. No error list is associated with this error number.

Recommended Action

None.

An example that causes this error number to return is if you have assigned the value ZSPI-VAL-ERR-AND-WARN to the token ZSPI-TKN-RESPONSE-TYPE (you want to see a response only if there is an error or warning), and no errors or warnings occurred during processing of the command. ZFUP-ERR-EMPTY-RESP indicates that all processing completed successfully.

14: ZFUP-ERR-NO-MEM

FUP ran out of internal memory space while trying to execute the command.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NO-MEM (14). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

FUP ends abnormally. An internal software error might have occurred. Enter the command again with the INSPECT SAVEABEND RUN option to produce a saveabend file. Contact your service provider and provide all relevant information:

- Saveabend file
- History of all commands and tokens sent to FUP and the tokens returned by FUP since FUP was started (as comprehensive as possible)
- Description of the problem and accompanying symptoms
- Details from the message or messages generated
- Supporting documentation (such as EMS logs, trace files, and a processor dump)

If local operation procedures require contacting the Global Customer Support Center (GCSC), supply your system number and the numbers and versions of related products.

15: ZFUP-ERR-EDITREAD

A command failed with an error from the EDITREAD or EDITREADINIT procedure.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list owned by ZSPI-SSN-ZGP1
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-EDITREAD (15). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the name of the file for which the error occurred.

`ZCOMMAND`

is the FUP command that was executing.

`ZOBJECT`

is the object type.

Recommended Action

The second error list describes an EDITREAD or EDITREADINIT procedure (ZSPI-SSN-ZGP1) error. To determine the cause of the error, enter this second error list. For details on extracting tokens from nested error lists, see Handling FUP Errors on page 2-17.

16: ZFUP-ERR-SORT

The command (LOAD or LOADALTFILE) failed with an error from the FASTSORT utility.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list owned by ZSPI-SSN-ZSRT
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SORT (16). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the name of the file on which FASTSORT was processing.

`ZCOMMAND`

is the FUP command that was executing.

`ZOBJECT`

is the object type.

Recommended Action

The second error list describes a FASTSORT (ZSPI-SSN-ZSRT) error. You must enter this second error list to determine the cause of the error. For details about extracting tokens from nested error lists, see Handling FUP Errors on page 2-17.

17: ZFUP-ERR-FILESYS

The command failed with a Guardian file-system error.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list owned by ZSPI-SSN-ZFIL
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-FILESYS (17). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file on which the error occurred.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The second error list describes a NonStop Kernel file-system (ZSPI-SSN-ZFIL) error. To determine the cause of the error, enter this second error list. For details on extracting tokens from nested error lists, see Handling FUP Errors on page 2-17.

18: ZFUP-ERR-GUARD

The command failed with a NonStop Kernel error.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list owned by ZSPI-SSN-ZGRD
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-GUARD (18). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with the fields:

ZNAME

is the name of the file on which the error occurred.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The second error list describes a NonStop Kernel (ZSPI-SSN-ZGRD) error. To determine the cause of the error, enter this second error list. For details about extracting tokens from nested error lists, see Handling FUP Errors on page 2-17.

19: ZFUP-ERR-SPI

The command failed with a Subsystem Programmatic Interface (SPI) error.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-ERRLIST        token-type ZSPI-TYP-LIST.
    Error list owned by ZSPI-SSN-ZSPI
  ZSPI-TKN-ENDLIST        token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SPI (19). This token is always in the error list.

Recommended Action

FUP ends abnormally. An internal software error might have occurred.

The second error list describes an SPI (ZSPI-SSN-ZSPI) error. To determine the cause of the error, enter this second error list. For details about extracting tokens from nested error lists, see Handling FUP Errors on page 2-17.

Enter the command again with the INSPECT SAVEABEND RUN option to produce a saveabend file. Contact your service provider and provide all relevant information:

- Saveabend file
- History of all commands and tokens sent to FUP and the tokens returned by FUP since FUP was started (as comprehensive as possible)
- Description of the problem and accompanying symptoms
- Details from the message or messages generated
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and a processor dump, if applicable

If your local operation procedures require contacting the Global Customer Support Center (GCSC), supply your system number and the numbers and versions of all related products as well.

20: ZFUP-ERR-PE

An internal programming error occurred.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
  ZFUP-TKN-PE-NUM         token-type ZSPI-TYP-INT
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-PE (20). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that could not be found.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-PE-NUM

is an integer that indicates where the programming error occurred.

Recommended Action

FUP ends abnormally. An internal software error might have occurred.

Enter the command again with the INSPECT SAVEABEND RUN option to produce a saveabend file. Contact your service provider and provide all relevant information:

- Saveabend file
- History of all commands and tokens sent to FUP and the tokens returned by FUP since FUP started (as comprehensive as possible)
- Description of the problem and accompanying symptoms
- Details from the message or messages generated
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and a processor dump, if applicable

If your local operation procedures require contacting the Global Customer Support Center (GCSC), supply your system number and the numbers and versions of all related products as well.

21: ZFUP-ERR-BAD-KEY

An application specified a LOADALTFILE command to load an existing alternate-key file, but the file had invalid alternate-key parameters.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZFUP-TKN-ALTFILE-NUM      token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BAD-KEY (21). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the primary file.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-ALTFILE-NUM

is the alternate-key file number.

Recommended Action

The command fails. To execute the LOADALTFILE command, correct the alternate-key parameters in the existing file.

22: ZFUP-ERR-BAD-PARTS

An application specified a command with invalid partition parameters. FUP found the file that it expected to be the primary file, but the file was either not a partitioned file or was the primary partition for the destination file.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BAD-PARTS (22). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Correct either the destination file name or the ZPARTOF volume name and retry the command.

23: ZFUP-ERR-BAD-TAPELABEL

An application specified a command with a TAPE DEFINE that does not match the actual tape label. (The tape label might be invalid.) The attempt to open the labeled-tape file failed with file-system error 196.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BAD-TAPELABEL (23). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Verify that the TAPE DEFINE attributes match the label for the tape file. Correct the attributes (if necessary) and retry the command.

24: ZFUP-ERR-BAD-VAR-BLOCKLEN

An application specified an invalid block length for a file with variable-length record blocking. An attempt to load the file with variable-length records failed because the length of a block was one or zero bytes. This error can occur if the file to be loaded was filled without specifying the VAROUT option for the FUP COPY command. Also, the data in the file might be corrupt.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZFUP-TKN-BLOCKLEN         token-type ZSPI-TYP-INT2.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BAD-VAR-BLOCKLEN (24). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-BLOCKLEN

is the invalid block length.

Recommended Action

Variable-length blocks must contain at least two bytes of length information at the beginning of the block. Correct the block (or re-create the file if necessary) and retry the command.

25: ZFUP-ERR-BAD-VAR-RECLEN

An application specified an invalid variable record length for a file. An attempt to load the file failed because the record length specified for a variable-length record was invalid (for example, the record length was a negative number). This error can occur if the file to be loaded was filled without specifying the VAROUT option of the FUP COPY command. Also, data in the file might be corrupt.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZFUP-TKN-RECLEN           token-type ZSPI-TYP-INT2.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BAD-VAR-RECLEN (25). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-RECLEN

is the invalid record length.

Recommended Action

Correct the record length (or re-create the file if necessary) and retry the command.

26: ZFUP-ERR-BLOCKIN-CONFLICT

An application specified a TAPE DEFINE for the source file, but the TAPE DEFINE has a value for BLOCKLEN that conflicts with ZBLOCK-SIZE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BLOCKIN-CONFLICT (26). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Correct the BLOCKLEN or ZBLOCK-SIZE value and retry the command.

27: ZFUP-ERR-BLOCKLEN-BIG

An application specified a TAPE DEFINE for the source file, but the TAPE DEFINE BLOCKLEN value is too large (greater than 32,767—the largest block size that FUP can process).

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
  ZFUP-TKN-BLOCKLEN       token-type ZSPI-TYP-INT2.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BLOCKLEN-BIG (27). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-BLOCKLEN

is the TAPE DEFINE BLOCKLEN value.

Recommended Action

The command fails. Change the BLOCKLEN value to a size that FUP can process, if possible, and retry the command.

28: ZFUP-ERR-DEFINE-CONFLICT

FUP attempted to alter a TAPE DEFINE BLOCKLEN or RECLEN value to match ZBLOCK-SIZE or ZRECORD-SIZE, but the FUP value is not a valid TAPE DEFINE value.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-DEFINE-CONFLICT (28). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Change the ZBLOCK-SIZE or ZRECORD-SIZE value, if possible, and retry the command.

30: ZFUP-ERR-AKNOUP

FUP did not change the alternate-key files.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-AKNOUP (30). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. No corrective action is needed. After the duplication finishes, update the alternate-key files using the LOADALTFILE command.

32: ZFUP-ERR-DUP-SEC-PART

An application attempted to rename multiple partitions to the same volume name in a DUPLICATE command rename options.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-DUP-SEC-PART (32). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file on which FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Each partition of a partitioned file must reside on a separate disk volume. Correct the rename options for the DUPLICATE command and retry the command.

33: ZFUP-ERR-EBCDICIN-CONFLICT

An application specified a TAPE DEFINE for the source file, but the TAPE DEFINE value for EBCDIC conflicts with ZEBCDIC; for example, if the EBCDIC value in the TAPE DEFINE was OFF, and ZEBCDIC was set to ZSPI-VAL-TRUE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-EBCDICIN-CONFLICT (33). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Correct the EBCDIC value in the TAPE DEFINE, or omit ZEBCDIC and retry the command.

34: ZFUP-ERR-SEC-PART

An application specified a secondary partition name in a context where secondary partitions are not allowed.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SEC-PART (34). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the file that FUP was processing.

`ZCOMMAND`

is the command that caused the error.

`ZOBJECT`

is the object type.

Recommended Action

Set the ZPART-ONLY field, if applicable, and retry the command.

35: ZFUP-ERR-EMPTY-SOURCE

An application specified an empty source file (zero records) for a LOAD command but did not set ZEMPTYOK to ZSPI-VAL-TRUE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-EMPTY-SOURCE (35). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

If the source file is empty and you want an empty destination file, set ZEMPTYOK to ZSPI-VAL-TRUE.

36: ZFUP-ERR-ENSURE-PARTS

An application attempted to duplicate a partition with a new extent size with ZPART-ONLY set to ZSPI-VAL-TRUE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-ENSURE-PARTS (36). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Although this is only a warning message, ensure that the extent sizes in the file label of the primary partition reflect the actual extent sizes of the secondary partitions.

37: ZFUP-ERR-FILE-KEY-INCOM

The alternate-key file is incompatible with the alternate keys. The existing alternate-key file characteristics contradict the keys defined in the primary-key file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZFUP-TKN-ALTFILE-NUM      token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-FILE-KEY-INCOM (37). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the primary file.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-ALTFILE-NUM

is the alternate-key file number.

Recommended Action

The command fails. Alter the ALTKEYs and ALTFILEs in the primary file.

38: ZFUP-ERR-IGN-COMPACT

An application attempted to LOAD a nonrelative file with ZCOMPACT set to ZSPI-VAL-TRUE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-IGN-COMPACT (38). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

This is a warning message only. No corrective action is needed. FUP ignores ZCOMPACT for nonrelative files.

40: ZFUP-ERR-IGN-RENAME-OPTS

An application attempted to duplicate a file with ZDEST set to ZFUP-VAL-OLD with either the rename or extent size option specified. These options are mutually exclusive.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-IGN-RENAME-OPTS (40). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

If you specify the rename or extent size options, you cannot set ZDEST to ZFUP-VAL-OLD.

41: ZFUP-ERR-INCOMPAT-FILE

An application specified a DUPLICATE command with incompatible source and destination files.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INCOMPAT-FILE (41). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. You cannot set ZDEST-OPTION to ZFUP-VAL-OLD for incompatible files. Purge the old file and retry the command.

42: ZFUP-ERR-INCON-PARTS

The file attributes of the individual partitions of a partitioned file are inconsistent with each other.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INCON-PARTS (42). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Check the file label partition attributes for all partitions and correct any inconsistencies. The file type, record length, data-block length, key length, key offset, and index-block length attributes must be the same.

For relative and entry-sequenced files, the partition extent sizes must be the same as those specified when the primary partition was created.

43: ZFUP-ERR-INV-FTYPE

An application attempted to load an unstructured file, which is not allowed for this file type.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-INV-FTYPE (43). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the name of the file that FUP was processing.

`ZCOMMAND`

is the FUP command that was executing.

`ZOBJECT`

is the object type.

Recommended Action

The command fails, and no recovery is possible. You cannot load unstructured files.

44: ZFUP-ERR-MISS-ALTFILE

An application attempted to rename a nonexistent alternate-key file in a DUPLICATE command (the source file does not have the specified alternate-key file).

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZFUP-TKN-ALTFILE-NUM      token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-MISS-ALTFILE (44). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that could not be found.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-ALTFILE-NUM

is the alternate-key file number of the nonexistent alternate-key file.

Recommended Action

To rename the file, determine the correct rename options for the DUPLICATE command and retry the command. If you are renaming a destination file, FUP must create the file. ZDEST-OPTION must be either ZFUP-VAL-NEW or ZFUP-VAL-PURGE.

45: ZFUP-ERR-MISS-PART

An application attempted to rename a nonexistent partition file in a DUPLICATE command. (The source file does not have the specified partition.)

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
  ZFUP-TKN-PART-NO        token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-MISS-PART (45). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that could not be found.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-PART-NO

is the partition number of the nonexistent partition file.

Recommended Action

This is an informational message only. No corrective action is needed. However, to rename the file, determine the correct rename options for the DUPLICATE command and retry the command.

47: ZFUP-ERR-NO-ALT-FILE

An application specified an alternate-key file that does not exist in a LOADALTFILE command.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZFUP-TKN-ALTFILE-NUM     token-type ZSPI-TYP-INT.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NO-ALT-FILE (47). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the primary file.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-ALTFILE-NUM

is the alternate-key file number.

Recommended Action

You can use the LOADALTFILE command only for existing alternate-key files. To determine the alternate-key files for a specific file, execute a FUP INFO command with the DETAIL option.

48: ZFUP-ERR-NO-EXTSIZE

An application attempted to duplicate a relative or entry-sequenced file with a new extent size and ZPART-ONLY set to ZSPI-VAL-TRUE.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NO-EXTSIZE (48). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. You cannot specify the extent size if converting relative or entry-sequenced files with ZPART-ONLY set to ZSPI-VAL-TRUE. Resend the command without specifying an extent size.

49: ZFUP-ERR-NO-ZSVR

The labeled-tape server ($ZSVR) is not available. An attempt to open a tape file failed with file-system error 195.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NO-ZSVR (51). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Start $ZSVR (or have an operator start it) and retry the command.

50: ZFUP-ERR-NOT-ON-PARTF

An application attempted to duplicate a partitioned file with the ZDEST-OPTION set to ZFUP-VAL-OLD.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NOT-ON-PARTF (50). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. You cannot set ZDEST to ZFUP-VAL-OLD for partitioned files. You can purge the old file and retry the command with the partition rename token.

51: ZFUP-ERR-OP-REJECT

An operator rejected a request to mount a tape, and the attempt to open the tape file failed with a file-system error 194.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-OP-REJECT (51). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Determine why the operator rejected the tape mount request. Resolve the problem and retry the command.

52: ZFUP-ERR-PNAME-BAD

A partition name in the file label is missing a back-slash (\) or dollar sign ($).

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-PNAME-BAD (52). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file name that caused the error.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Correct the invalid name in the file label and retry the command.

53: ZFUP-ERR-PNAME-NOT-NET

A partition name in the file label cannot be transformed into the network form because the name is seven characters long.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-PNAME-NOT-NET (53). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the primary partition file.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Move the partition file to a volume with a name fewer than seven characters long. Alter the primary partition file name to the new volume name and retry the command.

54: ZFUP-ERR-MUST-REARRANGE-DATA

An application attempted to duplicate a partition with ZPART-ONLY set to ZSPI-VAL-TRUE. FUP determined that the partition data would be reshuffled among the various partitions.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-MUST-REARRANGE-DATA (54). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file on which FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails, and data must be rearranged among partitions. Resend the command with ZPART-ONLY set to ZSPI-VAL-FALSE and convert the file as a whole.

55: ZFUP-ERR-RECIN-CONFLICT

An application specified a TAPE DEFINE for the source file, but the TAPE DEFINE RECLEN value conflicts with ZRECORD-SIZE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-RECIN-CONFLICT (55). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Correct the ZRECORD-SIZE or TAPE DEFINE RECLEN value and retry the command.

56: ZFUP-ERR-RECLEN-BIG

An application specified a TAPE DEFINE for the source file, but the TAPE DEFINE RECLEN value is too large (greater than 4096).

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
  ZFUP-TKN-RECLEN         token-type ZSPI-TYP-INT2.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-RECLEN-BIG (56). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-RECLEN

is the TAPE DEFINE RECLEN value.

Recommended Action

The command fails. Change the RECLEN value (if possible) to a size that FUP can process (less than 4096 bytes) and retry the command.

57: ZFUP-ERR-PART-KEY-LONG

A partial key is too long (greater than 255 bytes).

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-PART-KEY-LONG (57). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

The file label for the indicated file might be invalid. Check the partition partial key information for the file and correct it (if necessary).

58: ZFUP-ERR-RELOAD-ALTFILES

An application duplicated a relative or entry-sequenced file from one volume type to another volume type, and the record positions in the alternate-key files have changed.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-RELOAD-ALTFILES (58). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

Recommended Action

After the duplication completes, reload the alternate-key files using the LOADALTFILE command.

59: ZFUP-ERR-SHORT-KEYS

An application specified the LOADALTFILE command, but a number of records contain incomplete alternate-key fields. FUP does not generate them in the destination file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZFUP-TKN-ALTFILE-NUM      token-type ZSPI-TYP-INT.
  ZFUP-TKN-REC-COUNT        token-type ZSPI-TYP-INT4.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token consisting of the FUP subsystem ID and the error number ZFUP-ERR-SHORT-KEYS (59). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the primary file.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

ZFUP-TKN-ALTFILE-NUM

is the alternate-key file number.

ZFUP-TKN-REC-COUNT

is the number of records with incomplete keys.

Recommended Action

Pad the records in the primary file (so that alternate keys are completely contained in the records) and retry the command.

60: ZFUP-ERR-SOURCEDATE-NOT-SAVED

An application executed a DUPLICATE command, but FUP did not preserve the source date for the destination file. An attempt was made to duplicate a file to a remote system that is running an operating system older than B00.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SOURCEDATE-NOT-SAVED (60). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. No corrective action is needed.

62: ZFUP-ERR-TRUNC

FUP is truncating the input records of the current file.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-TRUNC (62). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. Ignore the message if truncation is intended, or check the ZBLOCK-SIZE, ZREC-SIZE and input record-length values if it is unintentional.

63: ZFUP-ERR-UNSTR-NO-EXTSIZE

An application specified the extent size when converting partitioned unstructured files, but this is not allowed.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-UNSTR-NO-EXTSIZE (63). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. Retry the command without specifying an extent size.

64: ZFUP-ERR-USE-EXT-N-READ

An application specified a TAPE DEFINE for an input file (read access) with the USE attribute set to EXTEND.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-USE-EXT-N-READ (64). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

You cannot open a tape file for read access using a TAPE DEFINE with the USE attribute set to EXTEND. Correct the TAPE DEFINE (or command parameters) and retry the command.

65: ZFUP-ERR-USE-OUT-N-READ

An application specified a TAPE DEFINE for an input file (read access) with the USE attribute set to OUT.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-USE-OUT-N-READ (65). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The command fails. You cannot open a tape file for read access using a TAPE DEFINE with the USE attribute set to OUT. Correct the TAPE DEFINE (or command parameters) and retry the command.

66: ZFUP-ERR-VAR-TRUNC

An application specified a LOAD command with ZVAR-REC set to ZSPI-VAL-TRUE, and FUP truncated the last variable-length record in the block. This record extended beyond the end of the block (according to the record length at the beginning of the record).

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-VAR-TRUNC (66). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that FUP was processing.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Although this is only a warning message, you need to ensure that the ZBLOCK-SIZE value (if specified) is correct and that the file was generated by FUP COPY with the VAROUT option.

67: ZFUP-ERR-VOL-NOT-FOUND

A partition or alternate-key file name points to a nonexistent volume.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-VOL-NOT-FOUND (67). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the primary file name.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Ensure that all of the partition names specified in FUP-DDL-PART-RENAME-OPTS (or the alternate-key file names that are specified in ZFUP-DDL-ALT-RENAME-OPTS) reside on existing volumes.

68: ZFUP-ERR-AUDITED-FILE

An application attempted a CHECKSUM command on an audited file. This is not allowed.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-AUDITED-FILE (68). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that caused the error.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

The CHECKSUM command fails on audited files. Use the CHECKSUM command only on nonaudited files.

69: ZFUP-ERR-SKIPIN-CONFLICT

An application attempted to use the ZSKIP option when the source file for the operation was a TAPE DEFINE (with the LABELS attribute set to label processing).

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SKIPIN-CONFLICT (69). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

The command fails. The ZSKIP option is not allowed for labeled tapes. Use the TAPE DEFINE FILESEQ attribute to skip files on a labeled tape.

70: ZFUP-ERR-REELS-CONFLICT

An application attempted to use the ZREELS option when the source file for the operation was a TAPE DEFINE (with the LABELS attribute set to label processing).

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-REELS-CONFLICT (70). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

The command fails. The ZREELS option is not allowed for labeled tapes. Use the TAPE DEFINE REELS and VOLUME attributes with multiple labeled tapes.

71: ZFUP-ERR-CORRUPT-FILE

An application executed a DUPLICATE command, but FUP determined that a file is corrupt.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-CORRUPT-FILE (71). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. No corrective action is needed. The indicated file is duplicated, and the corrupt flag is set.

72: ZFUP-ERR-BROKEN-FILE

An application executed a DUPLICATE command, but FUP determined that a file is broken.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-BROKEN-FILE (72). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. No corrective action is needed. The indicated file is duplicated, and the broken flag is set.

73: ZFUP-ERR-SAFEGUARD-LOST

An application executed a DUPLICATE command for a file with Safeguard protection, but the new file does not have Safeguard protection.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SAFEGUARD-LOST (73). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. No corrective action is needed. Use the Safeguard command interpreter (if necessary) to establish Safeguard protection for the new file.

74: ZFUP-ERR-CONVERT-CORRUPT

An application attempted to duplicate a corrupt file between different disk process types (DP1 or DP2), but this operation is not allowed.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-CONVERT-CORRUPT (74). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

The command fails. You cannot duplicate a corrupt file to a different disk process type.

75: ZFUP-ERR-NO-DEFINE

An application specified a TAPE DEFINE for the source file, but FUP could not find the DEFINE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NO-DEFINE (75). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

If the DEFINE is not present, add it using the TACL ADD DEFINE command. If the DEFINE name is incorrect in the ZFUP-TKN-SOURCE-FILE token, correct the name and retry the command.

76: ZFUP-ERR-KEPT

An application executed a DUPLICATE command with ZDEST-OPTION set to ZFUP-VAL-KEEP. FUP did not duplicate the file because the destination file name matched the source file name.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-KEPT (76). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

This is an informational message only. No corrective action is needed.

77: ZFUP-ERR-ALTKEY-LEN0

An application specified an alternate key with an invalid length of zero (0).

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-ALTKEY-LEN0 (77). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Correct the length of the alternate key. The file label for the indicated file might also be invalid. Check the alternate-key file descriptions for the file and correct them if necessary.

78: ZFUP-ERR-ALTKEY-LONG

An alternate key extends beyond the length of the record.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-ALTKEY-LONG (78). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the file that FUP was processing.

`ZCOMMAND`

is the command that caused the error.

`ZOBJECT`

is the object type.

Recommended Action

Correct the length of the alternate key. The file label for the indicated file might also be invalid. Check the alternate-key file descriptions for the file and correct them if necessary.

79: ZFUP-ERR-ALTFILE-PRIKEY-LONG

A primary key is too long for an alternate-key file.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-ALTFILE-PRIKEY-LONG (79). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

The combined length of the specified primary key, alternate key, and two-byte key must not exceed 255 bytes. Check the alternate-key file descriptions for the file and correct the lengths if necessary.

80: ZFUP-ERR-UNIQUE-N-NON-UNIQUE

Both unique and nonunique keys were specified in an alternate-key file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-UNIQUE-N-NON-UNIQUE (80). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Check the alternate-key file descriptions for the file and correct the keys if necessary.

81: ZFUP-ERR-VARYING-UNIQUE-ALT-KEYS

Unique keys were not the same length in the same alternate-key file.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-VARYING-UNIQUE-ALT-KEYS (81). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Unique keys with different lengths must reside in different alternate-keys files. Check the alternate-key file descriptions for the file and correct them if necessary.

82: ZFUP-ERR-KEYLEN-ZERO

A primary key length is zero, which is invalid.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-KEYLEN-ZERO (82). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Specify a nonzero length for the primary key and retry the command.

83: ZFUP-ERR-PART-KEY-MISSING

A partial key for a key-sequenced file partition is missing.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-PART-KEY-MISSING (83). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Specify the missing partial key and retry the command.

84: ZFUP-ERR-ALT-FILE-MISSING

An alternate key does not have a corresponding alternate-key file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-FILE-KEY-MISSING (84). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Correct the alternate key or alternate-key file and retry the command.

85: ZFUP-ERR-ALT-KEY-MISSING

An alternate-key file is missing a corresponding alternate key.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-ALT-KEY-MISSING (85). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file that FUP was processing.

ZCOMMAND

is the command that caused the error.

ZOBJECT

is the object type.

Recommended Action

Correct the alternate key or alternate-key file and retry the command.

86: ZFUP-ERR-NOT-ON-OPTICAL

An application specified an optical-disk file in a FUP command (such as LOAD or CHECKSUM) that does not support optical disks.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NOT-ON-OPTICAL (86). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file FUP was processing.

ZCOMMAND

is the FUP command number that caused the error.

ZOBJECT

is the object type.

Recommended Action

The command terminates. If you intended to specify another file that is not on an optical disk, correct the file specification and retry the command.

89: ZFUP-ERR-MUST-SQL-RECOMPILE

A DUP command duplicated a file sensitive to SQL.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-MUST-SQL-RECOMPILE (89). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file FUP was processing.

ZCOMMAND

is the FUP command number that caused the error.

ZOBJECT

is the object type.

Recommended Action

Although the DUPLICATE command was successful, the new file should be compiled with SQL.

90: ZFUP-ERR-NOT-ON-SQL-OBJECT

An application specified an SQL object in a FUP command that does not allow SQL objects.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NOT-ON-SQL-OBJECT (90). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file FUP was processing.

ZCOMMAND

is the FUP command number that caused the error.

ZOBJECT

is the object type.

Recommended Action

The command fails. If you intended to specify a file other than an SQL object, correct the file name and retry the command. Use the SQL conversational interface (SQLCI) to perform operations on SQL objects.

91: ZFUP-ERR-ALT-NOT-SQL

An application specified a LOADALTFILE command with an SQL object as an alternate-key file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-ALT-NOT-SQL (91). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file FUP was processing.

ZCOMMAND

is the FUP command number that caused the error.

ZOBJECT

is the object type.

Recommended Action

The operation terminates. Specify a valid alternate-key file and retry the command.

92: ZFUP-ERR-NOT-ON-VIEW

An application specified an SQL shorthand view or protection view in a CHECKSUM command.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NOT-ON-VIEW (92). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the file FUP was processing.

ZCOMMAND

is the FUP command number that caused the error.

ZOBJECT

is the object type.

Recommended Action

FUP does not perform the CHECKSUM operation on the view, but it continues with the next file or object if the application specified a file-set template. If you intended to specify another type of file (other than the view), correct the file specification and retry the command.

93: ZFUP-ERR-OUT-IS-SQL

An application specified an SQL object as an OUT (or list) file in the FUP startup message.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-OUT-IS-SQL (93). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the file FUP was processing.

`ZCOMMAND`

is the FUP command number that caused the error.

`ZOBJECT`

is the object type.

Recommended Action

The command terminates. Specify a valid OUT file and retry the command.

94: ZFUP-ERR-PART-IS-SQL

An application specified a DUPLICATE command for a partitioned file (with ZPART-ONLY set to ZSPI-VAL-FALSE), and one of the partitions was an SQL object.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-PART-IS-SQL (94). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the file FUP was processing.

`ZCOMMAND`

is the FUP command number that caused the error.

`ZOBJECT`

is the object type.

Recommended Action

None of the partitioned file is duplicated. If the application specified a file set, FUP continues with the next file in the file set. If you intended to specify another type of file, correct the file specification and retry the command.

100: ZFUP-ERR-REST-TOOMANY-FILES

An application specified more than one source file for the ZFUP-TKN-SOURCE-FILE token for a DUPLICATE command (with the restartable option specified).

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-REST-TOOMANY-FILES (100). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the source file template that caused the error.

ZCOMMAND

is the number of the command that caused the error. Its value is ZFUP-CMD-DUPLICATE (2).

ZOBJECT

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

Execute the DUPLICATE command (without the restartable option) to duplicate more than one source file with a single command. If you require the restartable option, execute a DUPLICATE command for each source file.

101: ZFUP-ERR-OPTICAL-RESTART-FILE

An application specified a restart file that is located on an optical disk volume for a DUPLICATE command (with the restartable option specified)—or for a RESTART command.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-OPTICAL-RESTART-FILE (101). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the name of the restart file.

`ZCOMMAND`

is the number of the command that caused the error. Its value is ZFUP-CMD-DUPLICATE (2) or ZFUP-CMD-RESTART (5).

`ZOBJECT`

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

Either retry one of the commands with a valid restart file that is not on an optical disk volume, or specify a default restart file on the default subvolume of the application (if it is not on an optical disk volume). Specify the default restart file as:

- For the DUPLICATE command, set ZFUP-TKN-RESTART-FILE to blanks and allow FUP to create the default restart file ZZRSTART.
- For the RESTART command, set ZFUP-TKN-FILE to blanks (or omit this token from the command buffer) and allow FUP to search for the default restart file ZZRSTART.

103: ZFUP-ERR-SRC-FILE-CHANGED

An application attempted a RESTART command, but the source file was modified since the original DUPLICATE command was issued. FUP checks the time of the last modification in the file label to determine if the source file was modified.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-SRC-FILE-CHANGED (103). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the source file that was modified.

ZCOMMAND

is the number of the command that caused the error. Its value is ZFUP-CMD-RESTART (5).

ZOBJECT

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

You cannot restart the duplicate operation for this source file. To start the operation from the beginning of the file, execute the DUPLICATE command again, with or without the restartable option.

104: ZFUP-ERR-DEST-NOT-CORRUPT

An application attempted a RESTART command, but the destination file is not corrupt.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-DEST-NOT-CORRUPT (104). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the name of the destination file.

`ZCOMMAND`

is the number of the command that caused the error. Its value is ZFUP-CMD-RESTART (5).

`ZOBJECT`

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

You cannot restart the DUPLICATE operation for this source file. Determine if the original DUPLICATE operation was successful. If it was not successful, execute the DUPLICATE command again (either with or without the restartable option) to start from the beginning of the source file.

105: ZFUP-ERR-REST-INFO-INVALID

An application attempted a RESTART command, but the RESTART file contains invalid information.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-REST-INFO-INVALID (105). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the restart file.

ZCOMMAND

is the number of the command that caused the error. Its value is ZFUP-CMD-RESTART (5).

ZOBJECT

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

You cannot restart the DUPLICATE operation for this source file. To start at the beginning of the source file, execute the DUPLICATE command again, with or without the restartable option.

106: ZFUP-ERR-DP-CHANGED

An application attempted a RESTART command, but the disk-process format of the volume containing the source or destination file was changed since the original DUPLICATE operation was executed.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-DP-CHANGED (106). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the source or destination file on the volume whose disk-process type was changed.

ZCOMMAND

is the number of the command that caused the error. Its value is ZFUP-CMD-RESTART (5).

ZOBJECT

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

Execute a DUPLICATE command without the restartable option. FUP converts the file to the new disk-process format.

107: ZFUP-ERR-NOT-REST-FILECODE

An application attempted a RESTART command, but the restart file does not have a file code of 855.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-NOT-REST-FILECODE (107). This token is always in the error list.

`ZFUP-MAP-CMD-ERROR`

is a structured token with these fields:

`ZNAME`

is the name of the restart file.

`ZCOMMAND`

is the number of the command that caused the error. Its value is ZFUP-CMD-RESTART (5).

`ZOBJECT`

is the object type. Its value is ZFUP-OBJ-FILE (1).

Recommended Action

Retry the RESTART command with a valid restart file (file code of 855), or execute the DUPLICATE command again (with or without the restartable option).

112: ZFUP-ERR-COMPACT

An empty RELATIVE file was found and not transferred during a FUP LOAD operation with ZCOMPACT set to ZSPI-VAL-TRUE.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZFUP-MAP-CMD-ERROR
    ZNAME                 type ZSPI-DDL-FNAME.
    ZCOMMAND              type ZSPI-DDL-INT.
    ZOBJECT               type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the FUP subsystem ID and the error number ZFUP-ERR-COMPACT (112). This token is always in the error list.

ZFUP-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the empty file.

ZCOMMAND

is the FUP command that was executing.

ZOBJECT

is the object type.

Recommended Action

Although this is an informational message only (and no corrective action is needed), the target file will have fewer records than the source file. The message is issued only once—when the first empty file is encountered.

C ORSERV Error Messages

The ORSERV process returns an error list in the response buffer when an error is detected in the command syntax or during execution of a command. The error lists are organized by error number (ZORS-ERR- value) in ascending order.

Each error-list description contains:

- A header displaying the numeric and symbolic values for the error
- A paragraph identifying the cause of the error
- A box listing the simple and structured tokens that can appear in the error list

  The notation used in the box for simple tokens is a shorthand version of the essential information given in the DDL TOKEN-CODE statement.

  Each error list begins with the ZSPI-TKN-ERRLIST token and ends with the ZSPI-TKN-ENDLIST token.

  Any nested error lists from other subsystems or software components (including the FASTSORT utility) also begin and end with these tokens.

  The remaining tokens are not in any particular order of occurrence.
- A description of each token in the error list (excluding ZSPI-TKN-ERRLIST and ZSPI-TKN-ENDLIST)
- A recommended course of action to correct or alleviate the error

The ORSERV process sets the return token ZSPI-TKN-RETCODE by the error number to identify the error list. If a command completes without an error, ORSERV sets ZSPI-TKN-RETCODE to ZORS-ERR-OK (0).

Although ZSPI-TKN-RETCODE is zero, the response buffer can still contain an error list. You need to investigate this warning. It signals a condition that does not prevent ORSERV from executing a command but could cause other problems.

To determine the problem, enter the error list and check the value of the ZSPI-TKN-ERROR token.

Note. For more information about ORSERV errors, see Section 4, ORSERV Programmatic Interface.

1: ZORS-ERR-INV-COMMAND

An application sent an invalid command to ORSERV.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-INV-COMMAND (1). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZCOMMAND

is the invalid command number.

ZNAME and ZOBJECT

are not used and have null values.

Recommended Action

Correct the invalid command in the SSINIT procedure call and retry the command.

2: ZORS-ERR-INV-OBJECT

An application specified an invalid object type for an ORSERV command.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-INV-OBJECT (2). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZOBJECT

is the invalid object type.

ZNAME and ZCOMMAND

are not used and have null values.

Recommended Action

The valid object type for all ORSERV commands is ZORS-OBJ-FILE. Correct the invalid object type and retry the command.

3: ZORS-ERR-INVALID-TOKEN

An application specified an invalid token in an ORSERV command.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-INVALID-TOKEN (3). This token is always in the error list.

ZSPI-TKN-PARM-ERR

is the standard SPI parameter error token, which contains these fields:

Z-TOKENCODE

is the token code that caused the error.

Z-INDEX and Z-OFFSET

are not used and have null values.

Recommended Action

The invalid token code appears in Z-TOKENCODE in the error list. Omit this token or use the correct token and retry the command.

4: ZORS-ERR-MISS-TOKEN

An application omitted a required token from an ORSERV command.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-MISS-TOKEN (4). This token is always in the error list.

ZSPI-TKN-PARM-ERR

is the standard SPI parameter error token, which contains these fields:

Z-TOKENCODE

is the missing token code that caused the error.

Z-INDEX and Z-OFFSET

are not used and have null values.

Recommended Action

The missing token code appears in Z-TOKENCODE in the error list. Add this token to the command buffer and retry the command.

5: ZORS-ERR-MISS-FIELD

A required field of a structured token has a null value.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-MISS-FIELD (5). This token is always in the error list.

`ZSPI-TKN-PARM-ERR`

is the standard SPI parameter error token, which contains these fields:

`Z-TOKENCODE`

is the token code that caused the error.

`Z-INDEX`

is the occurrence number of the token that caused the error.

`Z-OFFSET`

is the byte offset of the field that caused the error.

Recommended Action

Determine the field that has a null value from the Z-TOKENCODE and Z-OFFSET tokens in the error list. Supply a value for this field and retry the command.

6: ZORS-ERR-EXTRA-TOKEN

An application specified an extra token in an ORSERV command buffer.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-EXTRA-TOKEN (6). This token is always in the error list.

ZSPI-TKN-PARM-ERR

is the standard SPI parameter error token, which contains these fields:

Z-TOKENCODE

is the extra token code that caused the error.

Z-INDEX

is the occurrence number of the token that caused the error.

Z-OFFSET

is not used and has a null value.

Recommended Action

The extra token code appears in Z-TOKENCODE in the error list. Remove this token from the command buffer and retry the command.

7: ZORS-ERR-INV-VALUE

An application specified an invalid value for a token or for a field within a token.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-PARM-ERR
    Z-TOKENCODE           type ZSPI-DDL-TOKENCODE.
    Z-INDEX               type ZSPI-DDL-UINT.
    Z-OFFSET              type ZSPI-DDL-UINT.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

`ZSPI-TKN-ERROR`

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-INV-VALUE (7). This token is always in the error list.

`ZSPI-TKN-PARM-ERR`

is the standard SPI parameter error token, which contains these fields:

`Z-TOKENCODE`

is the token code of the token that caused the error.

`Z-INDEX`

is the occurrence number of the token that caused the error.

`Z-OFFSET`

is the byte offset of the field that caused the error. This token is null if the field is not part of a structured token.

Recommended Action

Determine the token or field that has an invalid value from the Z-TOKENCODE and Z-OFFSET tokens in the error list. Correct this value and retry the command.

8: ZORS-ERR-LONG-COMMAND

An application specified an ORSERV command that is too long for the command buffer.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-LONG-COMMAND (8). This token is always in the error list.

Recommended Action

Determine why the command is too long for the buffer. For example, check the length of the command buffer. The recommended buffer length for all ORSERV commands is ZORS-VAL-BUFLEN. Correct and retry the command.

9: ZORS-ERR-WRONG-SSID

An application specified an incorrect ORSERV subsystem ID (SSID) for an ORSERV command.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-WRONG-SSID (11). This token is always in the error list.

Recommended Action

Determine the correct ORSERV subsystem ID. Correct the SSINIT procedure call and retry the command.

10: ZORS-ERR-WRONG-SERVER

An application specified an ORSERV command that requires a newer version of the ORSERV server.

```
ZSPI-TKN-ERRLIST          token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR          token-type ZSPI-TYP-ERROR.
ZSPI-TKN-ENDLIST          token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-WRONG-SERVER (10). This token is always in the error list.

Recommended Action

You must use a newer version of ORSERV for this command. Determine the correct server version, start that version, and retry the command.

11: ZORS-ERR-SPI

The command failed with a Subsystem Programmatic Interface (SPI) error.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list with error returned by ZSPI-SSN-ZSPI
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-SPI (11). This token is always in the error list.

Recommended Action

ORSERV ends abnormally. An internal software error might have occurred.

The second error list describes an error reported by the SPI subsystem (ZSPI-SSN-ZSPI). An application must enter this error list to determine the cause of the error. For details about extracting tokens from nested error lists, see Handling ORSERV Errors on page 4-14.

Enter the command again with the INSPECT SAVEABEND RUN option to produce a saveabend file. Contact your service provider and provide all relevant information:

- Saveabend file
- The command and token values sent to ORSERV
- The token values returned by ORSERV in the response buffer
- Description of the problem and accompanying symptoms
- Details from the message or messages generated
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and a processor dump, if applicable

If your local operation procedures require contacting the Global Customer Support Center (GCSC), supply your system number and the numbers and versions of all related products as well.

12: ZORS-ERR-PE

An internal programming error occurred.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
  ZORS-TKN-PE-NUM           token-type ZSPI-TYP-INT
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-PE (12). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that ORSERV was processing.

ZCOMMAND

is the ORSERV command that was executing.

ZOBJECT

is the object type.

ZORS-TKN-PE-NUM

is an integer that indicates where the programming error occurred.

Recommended Action

Report this error to your Compaq service provider and provide:

- The command and token values you sent to ORSERV
- The token values returned by ORSERV in the response buffer
- A copy of the ORSERV server program file that returned the error
- A copy of the SAVE file, if ORSERV abended

13: ZORS-ERR-FILESYS

The ORSERV command failed with a Guardian file-system error.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list with error returned by ZSPI-SSN-ZFIL
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-FILESYS (13). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that ORSERV was processing.

ZCOMMAND

is the ORSERV command that was executing.

ZOBJECT

is the object type.

Recommended Action

The second error list contains an error reported by the Guardian file system (ZSPI-SSN-ZFIL). An application must enter this error list to determine the cause of the error.

For details on extracting tokens from nested error lists, see Handling ORSERV Errors on page 4-14.

14: ZORS-ERR-GUARD

The command failed with a NonStop Kernel error.

```
ZSPI-TKN-ERRLIST           token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR           token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                  type ZSPI-DDL-FNAME.
    ZCOMMAND               type ZSPI-DDL-INT.
    ZOBJECT                type ZSPI-DDL-INT.
  ZSPI-TKN-ERRLIST         token-type ZSPI-TYP-LIST.
    Error list with error returned by ZSPI-SSN-ZRD
  ZSPI-TKN-ENDLIST         token-type ZSPI-TYP-SSCTL.
ZSPI-TKN-ENDLIST           token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-GUARD (14). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZNAME

is the name of the file that ORSERV was processing.

ZCOMMAND

is the ORSERV command that was executing.

ZOBJECT

is the object type.

Recommended Action

The second error list contains an error reported by the NonStop Kernel (ZSPI-SSN-ZGRD). An application must enter this error list to determine the cause of the error.

For details about extracting tokens from nested error lists, see Handling ORSERV Errors on page 4-14.

15: ZORS-ERR-ORELOAD-INPROGRESS

An application specified the ZORS-CMD-ONLINERELOAD command for a file, but an online reload operation was already in progress for the file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-ORELOAD-INPROGRESS (15). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZCOMMAND

is the command number.

ZNAME

is the name of the target file.

ZOBJECT

is the object type.

Recommended Action

An application cannot initiate two online reload operations for the same file. If the reload was intended for another file, correct the command and retry it.

16: ZORS-ERR-NO-ORELOAD

An application specified the ZORS-CMD-SUSPEND command for a file, but an online reload operation had not been initiated for the file.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-NO-ORELOAD (16). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZCOMMAND

is the command number.

ZNAME

is the name of the target file.

ZOBJECT

is the object type.

Recommended Action

An application cannot suspend an online reload operation that has not been initiated.

17: ZORS-ERR-CANT-SUSPEND

An application specified the ZORS-CMD-SUSPEND command for a file, but the online reload operation cannot be suspended because:

- ORSERV called the KEYPOSITION file-system procedure for the ZZRELOAD file, but an error occurred on the call (for example, a NOWAIT operation has not yet completed for ZZRELOAD file).
- ORSERV called the STOP file-system procedure to stop the ORSERV process that is reloading the target file, but an error occurred on the STOP call.

```
ZSPI-TKN-ERRLIST            token-type ZSPI-TYP-LIST.
  ZSPI-TKN-ERROR            token-type ZSPI-TYP-ERROR.
  ZORS-MAP-CMD-ERROR
    ZNAME                   type ZSPI-DDL-FNAME.
    ZCOMMAND                type ZSPI-DDL-INT.
    ZOBJECT                 type ZSPI-DDL-INT.
ZSPI-TKN-ENDLIST            token-type ZSPI-TYP-SSCTL.
```

Tokens

ZSPI-TKN-ERROR

is the standard SPI error token, which consists of the ORSERV subsystem ID and the error number ZORS-ERR-CANT-SUSPEND (17). This token is always in the error list.

ZORS-MAP-CMD-ERROR

is a structured token with these fields:

ZCOMMAND

is the command number.

ZNAME

is the name of the target file.

ZOBJECT

is the object type.

Recommended Action

You cannot suspend the online reload operation for the file.

Index

F

G

H

I

L

M

N

O

P

R

S

T

U

W

Z

Special Characters

Contact NonStop Himalaya Publications

Mail, E-Mail, or Fax Your Comments

To ATTN: Product Manager—Software Publications

LOC CAC-03
Compaq Computer Corporation
NonStop Division
Cupertino, CA 95014-2599
pubs.comments@compaq.com

Fax: (408) 285-6230

Number of Pages
(include cover sheet): _____

From Name _____
Organization _____
Work phone _____
E-mail _____

What publication do you have a comment about? (Please provide complete title or part number.)

_____

Which software release and which *NonStop™ Himalaya* server model are you running?

_____

What page/section/topic of the publication do you have a comment about?

_____

Is your comment regarding (Check all that apply):

- ☐ An error in the publication?
- ☐ An omission in the publication?
- ☐ An general question or clarification about the publication?
- ☐ A problem with the TIM collection?
- ☐ A problem with the TIM viewer?

Please enter your comments in the space provided below:

_____
_____
_____
_____

If there is a problem, is it

- ☐ **Critical**: I cannot continue to work without immediate resolution.
- ☐ **Major**: I can continue to work, but prompt resolution is requested.
- ☐ **Minor**: I can continue to work, but eventual resolution is requested.

COMPAQ
NonStop Division

3. FUP Commands and Responses

4. ORSERV Programmatic Interface

5. ORSERV Commands and Responses

C. ORSERV Error Messages

Index

Examples

Figures

Tables