core Flight System (cFS)

File Manager (FM)

Version 2.5.2.0

APPLICATION USER'S GUIDE for Flight Operators

The cFS FM application provides onboard file system management services by processing commands for files and directories.

Flight Software Systems Branch – Code 582
Release Version 1.0 – 11/01/19
582-2014-001

National Aeronautics and
Space Administration

PREFACE

This is a generic, reusable guide. Tailor this guide for any mission by replacing text in this orange color with mission-specific information.

This core Flight System (cFS) File Manager (FM) Application User's Guide for Flight Operators provides operations guidance for the cFS FM Application.

This is one of a set of enhanced user's guides for the cFS Product Documentation Suite. While the main audience is flight operators, the guides also help serve the needs of flight software developers, Flight Software Sustaining Engineering (FSSE) and Integration and Test (I&T), and others who support missions that use cFS.

REVIEW/APPROVAL PAGE

This page applies before mission tailoring.

Prepared By:

Gary Smith Date
Technical Writer

Reviewed By:

Elizabeth Timmons Date
Associate Head, Flight Software Systems Branch (Code 582)
(Electronic Approvals available at: https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_fm

Dan Berry Date
Flight Software Engineer, Flight Software Systems Branch (Code 582)
(Electronic Approvals available at: https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_fm

Sandy Calder Date
Sustaining Engineer, Flight Software Systems Branch (Code 582)
(Electronic Approvals available at: https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_fm)

Mike Blau Date
Flight Software Engineer, Flight Software Systems Branch (Code 582)
(Electronic Approvals available at: https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_fm

Electronic signatures on file.

UPDATE HISTORY

Version	Date	Description	Affected Pages
Version 1.0	11/01/19	Initial Release	All

CONTENTS

TABLE OF FIGURES

TABLES

— GET DIRECTORY LISTING TELEMETRY PACKET —

— GET FILE INFO TELEMETRY PACKET —

— GET FREE SPACE TELEMETRY PACKET —

— GET OPEN FILES TELEMETRY PACKET —

— HOUSEKEEPING TELEMETRY PACKET —

— CONFIGURATION PARAMETERS —

— COMMANDS —

— ERROR EVENT MESSSAGES —

— INFO EVENT MESSAGES —

— DEBUG EVENT MESSSAGES —

Chapter 1.Introduction to this Document

1.1 *Purpose and Scope of this Guide*

This Application User's Guide is intended help Flight Operators and other users of the software (operations personnel, test engineers, and maintenance engineers) understand the cFS File Manager (FM) application.

- As delivered, this is a generic document ready to insert mission defined values to serve the needs of specific missions.
- The nomenclature of command and telemetry mnemonics is highly mission-specific. The mnemonics for specific missions will differ from the mnemonics in this Guide.
- This document does not address integration of FM into a cFS system.

1.2 *Acknowledgements*

This Application User's Guide relies heavily on the content of heritage FM publications and presentations, including the Doxygen-generated HTML user's guide and FM source code, and interviews with Flight Software (FSW) engineers. Thank you to the late Scott Walling, developer, and developers Dave Hardison, Susie Strege, Walt Moleski, and others who have built legacy code and guided us to understanding. This publication is a team effort. Thank you to the developers, the cFE/cFS and Code 582 management team, the Plankton, Aerosol, Cloud, ocean Ecosystem (PACE) mission that provided resources and comments, and the review team of Elizabeth Timmons, Dan Berry, Sandy Calder, and Mike Blau.

1.3 *Conventions and Terminology*

In this document:

- *Application* in this document refers to a set of data and functions that is treated as a single entity by the Core Flight Executive (cFE). The cFE resources are allocated on a per-application basis. The cFS FM applications is made up of a main task and one child task.
- *cFS File Manager application*, the *cFS FM application*, *cFS FM*, and *FM* are used interchangeably.
- *core Flight Executive* and *core Flight System* are abbreviated with a lower case "c" (cFE and cFS respectively).
- *Operating System Abstraction Layer (OSAL)* refers to the set of functions supplied as part of the cFS that isolate the calling application from operating system dependencies.
- While event names such as FM_STARTUP_TABLE_INIT_ERR_EID, for example, are used in the flight software and in this document, operators and users monitoring from ITOS or ASIST see the EID number instead (such as 6, in this example).

1.4 *Related Documents*

Documents used in the preparation of this Guide are listed in the table below.

Table 1-1. Related Documents

Item No.	Doc ID	Doc Type	Document Source
1	N/A	Design Presentation (PowerPoint)	Walling, Scott. *Core Flight System File Manager (FM) Application Design As Built.* Greenbelt, MD: Goddard Space Flight Center, Code 582 (Flight Software Branch).
2	N/A	Doxygen Compiled User Guide (HTML)	Kobe, David; Moleski, Walt; Strege, Susanne, et al. *CFS File Manager (FM) User's Guide.* Greenbelt: NASA Goddard Space Flight Center, Code 582, CFS Product Development Team, 04 April 2009.
3	582-2007-032	Requirements (Adobe PDF)	*CFS File Manager (FM) Requirements Document, Version 1.3.* Greenbelt: NASA Goddard Space Flight Center, Code 582, Flight Software Systems Branch, 1 August 2011.
4	582-2007-026	Heritage Analysis (Word)t	*Core Flight System (CFS) File Manager Application Heritage Analysis, Version 1.1.* Greenbelt: NASA Goddard Space Flight Center, Code 582, Flight Software Systems Branch, 16 August 2007.
6	582-2007-043	Developer Standards (Word)	David L. Kobe, the Hammers Company Inc. *Core Flight System (CFS) Development Standards Document, Version 1.3.* Greenbelt: NASA Goddard Space Flight Center, Code 582, Flight Software Systems Branch, 1 June 2012.
7	582-2007-001	Developer Guidance (Word)	Maureen O. Bartholomew, Code 582, David L. Kobe, the Hammers Company, Inc. *Core Flight Executive (cFE) Flight Software Application Developers Guide, Version 5.4*. Greenbelt: NASA Goddard Space Flight Center, Code 582, Flight Software Systems Branch, 9 July 2014.

1.5 *Assumptions*

1.5.1 Personnel

This Application User's Guide assumes the reader is a Flight Operator or is performing the equivalent role.

1.5.2 Software

The following list summarizes the assumptions made about cFS FM as documented in this Guide:

Source Code and Configuration

- FM has been configured using standard FM configuration parameters; the FM base code has not been modified.
- The cFE Application Programming Interface (API) and the OSAL are being used.
- The command mnemonics shown in this Guide are compatible with both Advanced Spacecraft Integration & System Test (ASIST) ground control software and Integrated Test and Operations System (ITOS).

Doxygen

- References in this document to Doxygen compiled hypertext markup language (HTML) user guide for developers and sustaining engineers assume that the HTML has been compiled from mission-specific source files.

1.6 *How to Use this Document*

Experienced flight operators may only need to browse this Guide, and use the Reference chapter as needed. New operators may wish to get more familiar with the entire Guide

Chapter 1 through Chapter 3 of this guide is intended as a learning tool, while Chapter 4 is intended as a reference tool.

Developers and sustaining engineers may wish to review the separate Doxygen-generated HTML user guide, or review the source files directly. Doxygen is the Code 582 standard tool for generating an on-line documentation browser in HTML from cFS and cFE source code and embedded developer comments. In contrast to this cFS FM Application User's Guide, the Doxygen compiled HTML user guide is primarily targeted to developers and sustaining engineers.

If searching this document for a particular event message that appears on the ground FSW (i.e., ASIST or ITOS) display, search using the English language portion of the message string. One and two digit sequential identifiers, and abbreviated English language nicknames and other elements from the source code are used for convenience within this document to organize and identify event messages, but in general, they are not visible while the application is running.

1.6.1 Providing Feedback

Readers who find an error in this guide or want to provide suggestions or other feedback can do so through the FM github issue tracker (https://github.com/nasa/fm/issues) or the generic ticket submission email address (gsfc-cfs-submitticket@mail.nasa.gov).

1.7 *Acronyms and Abbreviations*

Acronyms and abbreviations in this publication are shown in Table 1-2 below.

Table 1-2. Acronyms and Abbreviations

Acronym or Abbreviation	Description
API	Application Programming Interface
ASIST	Advanced Spacecraft Integration & System Test
BAT	Burst Alert Telescope

Acronym or Abbreviation	**Description**
CC	Command Code
cFE	Core Flight Executive
cFS	core Flight System
CI	Command Ingest
Cmd	Command
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DS	Data Storage
EID	Event Identifier
FM	File Manager
FSSE	Flight Software Sustaining Engineering
FSW	Flight Software
GNU	GNU's Not Unix
GSFC	Goddard Space Flight Center
HK	Housekeeping
ISIM	Integrated Science Instrument Module
ITOS	Integrated Test and Operations System
JWST	James Webb Space Telescope
LRO	Lunar Reconnaissance Orbiter
OS	Operating System
OSAL	Operating System Abstraction Layer
PACE	Plankton, Aerosol, Cloud, ocean Ecosystem
Pkt	Packet
RAM	Random-Access Memory
RDL	Record Definition Language
RTOS	Real Time Operating System
RTP	Relative Time Processor
SB	Software Bus Service
SBC	Single Board Computer
SC	Stored Commands task
SCH	Scheduler [Application]
SW	Software

Acronym or Abbreviation	Description
TBL	Table
TBL	cFS Table Services
TLM	Telemetry
TO	Telemetry Output
TOC	Table of Contents
UTC	Coordinated Universal Time

Chapter 2. **Introduction to cFS File Manager (FM)**

2.1 ***Heritage***

Development of the cFS File Manager (FM) application was first begun in 2007. FM derives its heritage from the LRO mission, which made use of the cFE and operating system API, and which divided file management and uplink/downlink services into two separate tasks.

2.2 ***cFS FM High Level Overview***

The cFS FM application provides onboard file system management services by processing ground commands for copying, moving, and renaming files; decompressing files; creating directories; deleting files and directories; providing file and directory informational telemetry messages and open file and directory listings; and assigning read/write file permissions.

2.2.1 Inputs to cFS FM

Inputs to cFS FM include:

- Inputs from the file system(s) via cFE OSAL
- Housekeeping (HK) request commands from a Scheduler (SCH) application
- Commands, which can originate from any source, e.g., command ingest (CI), stored command (SC), or other apps
- Commands received from the command pipe, which can originate from any source
- Free space table information from cFE TBL memory and the on-board file system via the cFE TBL application

2.2.2 Outputs from cFS FM

Outputs from cFS FM include:

- Table management information for the cFE TBL memory and the on-board file system, via the cFE TBL application
- Telemetry packets, such as command response packets (file info, directory list, free space, and open files) that are sent to wherever telemetry is sent (e.g., to the Data Storage (DS) and Limit Checker (LC) apps, events to HK, etc. Technically, these are sent on the software bus.
- Event messages, generated via cFE/EVS
- Outputs to the file system(s), via cFE OSAL

2.2.3 cFS FM Tables

The cFS FM application defines one table that specifies the file systems contained in the mission's system, as shown in Table 2-1 below.

The number of file systems for which FM can be enabled to report free space data is specified by the configuration parameter FM_TABLE_ENTRY_COUNT (see Table 4-55).

Table 2-1. Defining the File Systems Used

Record	Description
State	An integer value ranging from 0 to 2 indicating the state of this entry. The values correspond to "Unused", "Enabled", and "Disabled"
Name	The full path specification of the file system, e.g., "/ram".

2.2.4 cFS FM Software Context

Figure 1 below shows a typical software context for cFS FM.

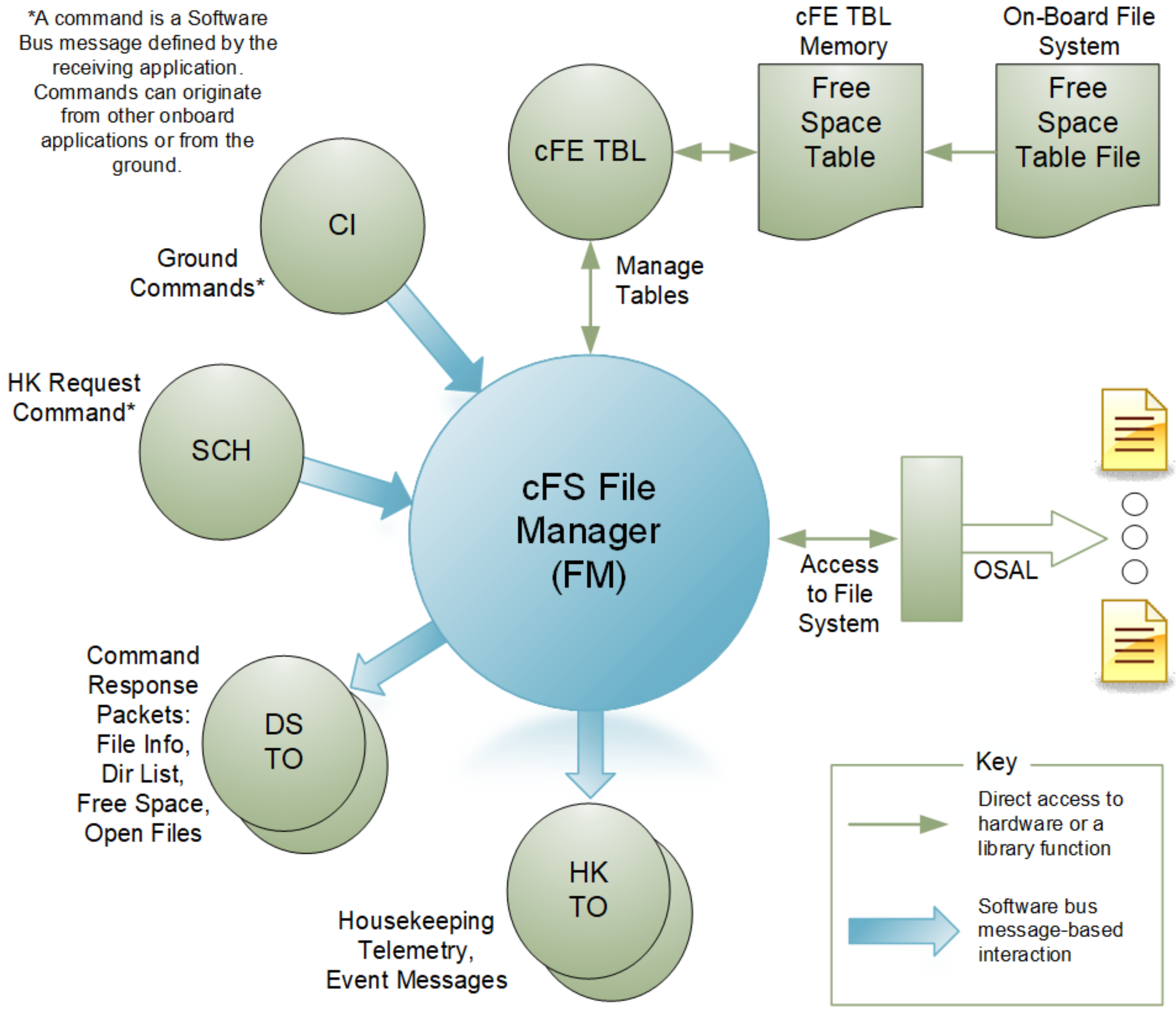
*A command is a Software Bus message defined by the receiving application. Commands can originate from other onboard applications or from the ground.
cFE TBL Memory
On-Board File System
cFE TBL
Free Space Table
Free Space Table File
CI
Ground Commands*
Manage Tables
HK Request Command*
SCH
cFS File Manager (FM)
Access to File System
OSAL
Command Response Packets: File Info, Dir List, Free Space, Open Files
DS TO
HK TO
Housekeeping Telemetry, Event Messages
Key
Direct access to hardware or a library function
Software bus message-based interaction

Figure 2-1. cFS FM Typical Software Context

The figure above shows a typical software context, with cFS system applications as configured for a particular mission communicating via the cFE Software Bus application with both cFS applications and mission-created applications that GSFC typically utilizes.

Table 2-2. Software Context Detail

Application	Software Context
cFE	When using FM (or any cFS application), the core Flight Executive (cFE) is required. Missions can use other cFS applications in the flight software system as well as mission specific cFS compliant applications.
cFE TBL	cFS FM learns of any ground updates to the cFS FM tables through the cFE TBL application.
CI	The FM application interfaces with a Command Ingest (CI) application to receive commands. See Section 4.3 for FM commands.
FM Housekeeping	See Section 4.1 for a detailed description of the FM housekeeping telemetry packet.
HK, TO, DS	HK , Telemetry Output (TO), and Data Storage (DS) applications, as configured for a particular mission, may subscribe to FM housekeeping telemetry packets.
OSAL	FM utilizes the OSAL API library functions for interfacing with the onboard file systems when processing commands.
SCH	The FM application interfaces with the cFS Scheduler (SCH) or other scheduling application for receiving periodic housekeeping requests.
Software Bus and System Apps	cFS System applications, as configured for a particular mission, may communicate with cFS FM via the cFE Software Bus (SB) application.

2.3 *File Types*

Certain cFS FM commands produce files when cFS FM processes the command. These files include a decompression file, a concatenation file, and a directory listing file, as described below in Sections 2.3.1, 2.3.2, and 2.3.3.

2.3.1 Decompression File

The decompression file is produced using the cFE API function CFE_FS_Decompress, which uses code ported from GNU zip sources. The cFE API function CFE_FS_Decompress produces the decompressed output file. The output file is comparable to that of a file that has been unzipped using the system utility *gunzip*. A compressed file reduces the size of the file by encoding the file. Compressed files may be used to reduce uplink time to the spacecraft. Decompressing the file will restore the file from its encoded state to the original state of the file.

2.3.2 Concatenation File

The concatenation file is a direct copy of the contents of the two command specified source files, with the contents of the first source file specified preceding the contents of the second source file specified.

The contents of the second source file are written with no delineation characters such as spaces or returns following the contents of the first source file.

2.3.3 Directory Listing File

The directory listing file is written as a binary file. The file contains a cFE file header at the top of the file contiguously followed by a directory listing status data structure. This data structure contains an echo of the command specified directory name. It also contains the following data points, each expressed as 32 bit unsigned integers:

- Directory size in bytes,
- Total number of files contained in the directory, and
- Number of file names written in the directory listing file.

The directory listing is then contiguously written to the file, one entry at a time. Each file in the directory contains a directory listing. Each entry in the directory listing includes:

- The name of the file,
- The file size in bytes (32 bit unsigned integer), and
- Last modification time of the file (32 bit unsigned integer).

2.4 *Converting File System Time to Spacecraft Time*

File systems use specific time epochs for their time tagging of files. Since spacecraft systems rarely use an epoch that matches a particular file system, a function is used to convert the file system time in seconds to spacecraft time in seconds. The conversion is controlled external to FM by a cFE configuration parameter which is set equal to the difference in the number of seconds between the spacecraft's epoch and the file system's epoch.

Chapter 3. **cFS FM Operation**

3.1 ***Initialization***

Upon a Power-On or Processor reset, the FM application initializes all telemetry within its housekeeping telemetry packet. The Command and Command Error counters are set to zero, as are all other numerical data within the packet. In addition to initializing telemetry, FM:

During initialization, FM performs the common cFE initialization (see Section 3.3.2), then loads the File System Free Space table, and creates the FM child task. Errors encountered during the initialization sequence will generally result in the termination of the FM application.

During initialization, FM must succeed in registering the File System Free Space table with cFE Table Services or the application will terminate.

FM will not terminate if it is unable to load the default table data file, although Command 16 - Get Free Space will fail until a valid table data file is loaded.

3.2 ***Normal Operations and Program Flow***

The FM application uses the typical cFS application format. At startup, FM performs an initialization sequence and then enters an infinite loop waiting for commands received via the cFE Software Bus, as shown in Figure 3-1 below.

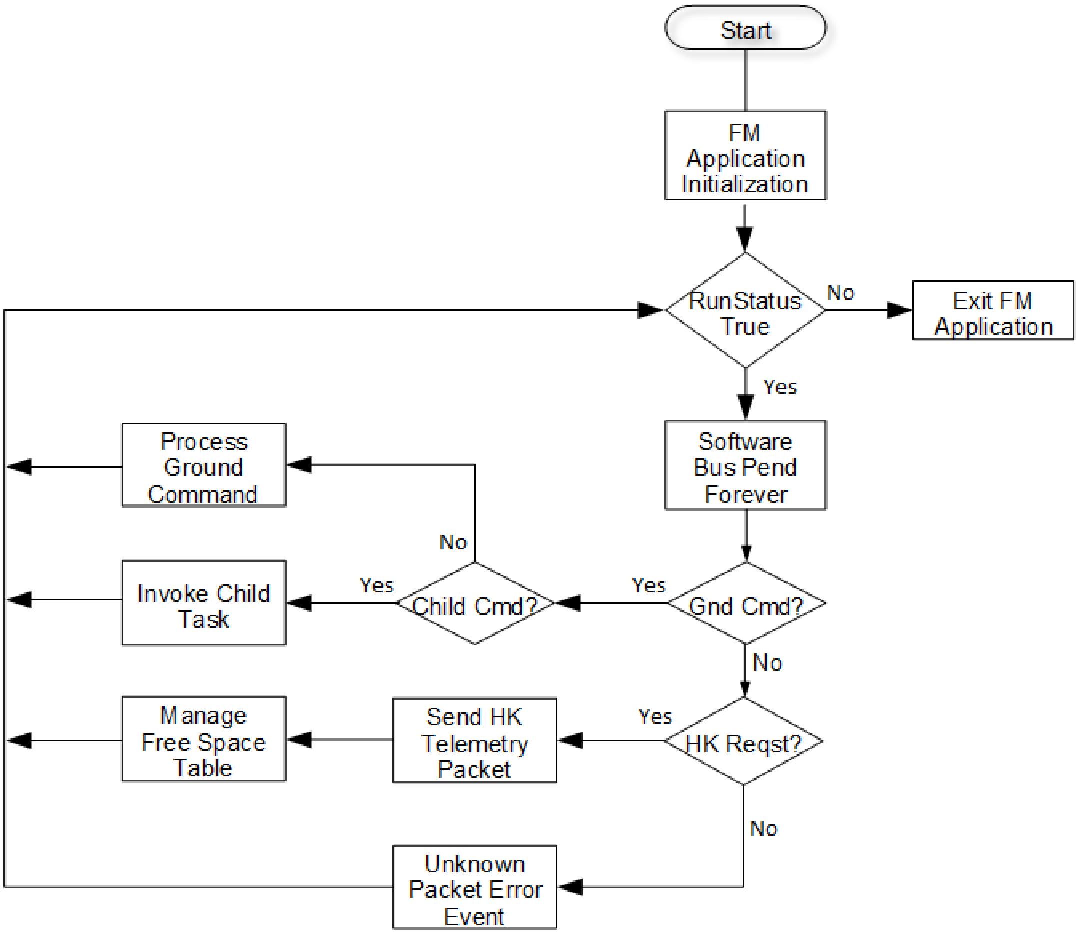

Figure 3-1. Main Task Flow

cFS FM is a command and telemetry driven application. cFS FM waits in a command loop infinitely until the software receives a scheduled housekeeping request, ground command, or stored command. Command packets (including housekeeping requests) are processed as they are received.

The application sends a cFS FM housekeeping telemetry message to subscribing applications upon receipt of a scheduled housekeeping request produced by the SCH application. This request is typically performed every four to five seconds. Depending on the configuration of the rest of the system, these cFS FM housekeeping telemetry messages may or may not be sent to the ground.

Various cFS FM ground commands produce telemetry messages when processing the command. These telemetry messages include:

- A directory listing telemetry message,
- A file information telemetry message, and
- An open file listing telemetry message.

3.2.1 FM Child Task

Figure 3-2 below shows the flow of an atypical feature, the FM child task. The cFS FM application uses the FM child task to process FM commands that might take a very long time to execute. It is also used for commands that take a variable duration to execute.

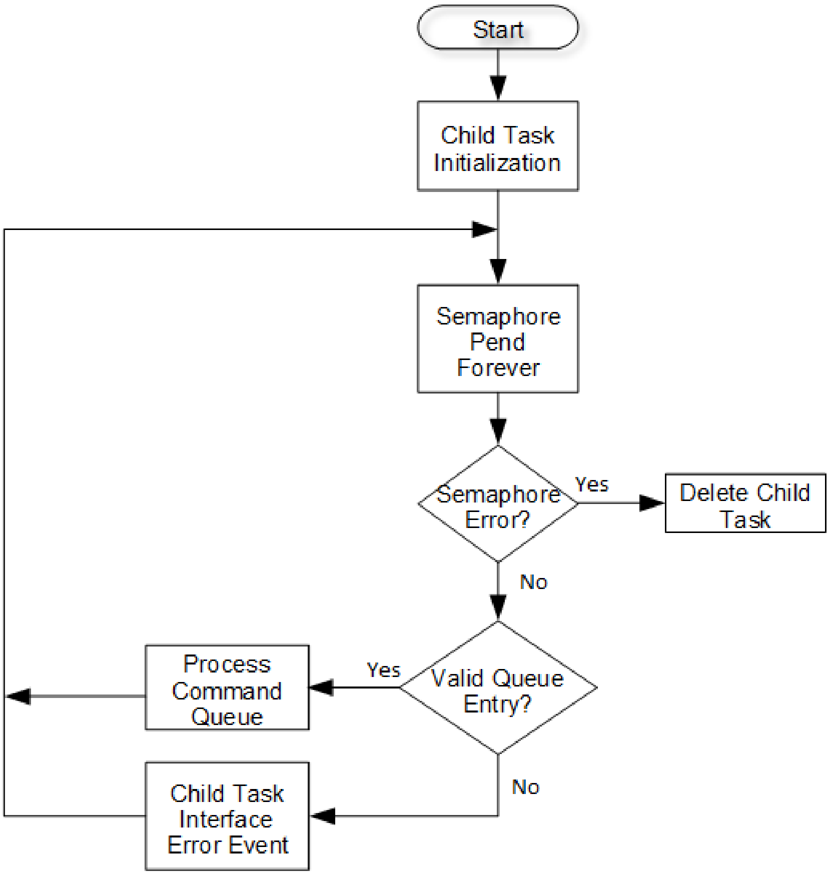

Figure 3-2. Child Task Flow

FM commands that process files which may be very large, and commands that process directories that have a very large number of entries can take several seconds to execute. This creates two potential issues: the length of time that FM would be unresponsive to new commands and the length of time that other lower priority tasks would be held off from execution.

Thus, creating a lower priority child task to process "slow" commands, or commands that take a variable duration to execute, immediately allows FM to be ready to receive the next command. In addition, inserting task delays into the child task command process loops provides other lower priority tasks with execution opportunities. Note that inserting delays into the main FM task would have lengthened the time that FM was unavailable for new commands.

If the child task cannot be created during FM application initialization, the FM application will not exit, but commands that rely on the child task to execute will fail.

Some initial event messages (after confirmation of command syntax) may be generated by the main FM task, while event messages related to command completion may be generated up to several seconds later by the child task.

The commands listed below pass processing to the child task.

- Concatenate Files
- Copy File
- Create Directory
- Decompress File
- Delete All Files

- Delete Directory
- Delete File
- Get Directory List (to file)
- Get Directory List (to packet)
- Get File Info
- Move File
- Rename File
- Set File Permissions

3.3 *Operational Constraints*

3.3.1 Files

All FM commands that input a file or directory name must use a full path specification. Any path specified must reference a directory configured in the Platform Support Package Volume Table. Other file constraints:

- The Decompress File Command should only be used on files that have been compressed.
- The contents of a directory must be deleted before the directory itself can be deleted.
- There are two levels of limits to the number of open files. The first limit is defined by the CFE-OSAL configuration element OS_MAX_NUM_OPEN_FILES and must be less than or equal to the second limit which is a value defined in a manner specific to the selected operating system.

  Note also that certain directory access functions (reading directory entries, etc.) require opening the directory and may affect the count of available open files.

3.3.2 cFE Services

The cFE provides a project-independent Flight Software (FSW) operational environment with a set of services that are the functional building blocks to create and host FSW Applications. The cFE is composed of six core services: Executive Service (ES), Software Bus Service (SB), Event Service (EVS), Table Service (TBL), File Service (FS), and Time Service (TIME). Each cFE service defines an API that is available to the application as a library of functions.

At start-up, cFS FM depends on cFE services for the following:

- Starting the application and child task with the cFE ES application;
- Registering tables with the cFE TBL application;
- Subscribing to the cFE Event Services application; and
- Creating pipes and subscribing with the cFE Software Bus application.

While running, cFE, in turn, is dependent on the cFE Software Bus Receive API.

Chapter 4. **cFS FM Reference**

This chapter details telemetry, configuration parameters, commands, and event messages.

4.1 ***Telemetry***

FM generates five telemetry packets:

1. Get Directory Listing Entry Packet (see Section 4.1.1)
2. Get File Info Telemetry Packet (see Section 4.1.2)
3. Get Free Space Telemetry Packet (see Section 4.1.3)
4. Get Open Files Telemetry Packet (see Section 4.1.4)
5. Housekeeping Telemetry Packet (see Section 4.1.5)

4.1.1 Get Directory Listing Telemetry Packet

The Get Directory Listing Telemetry Packet contains five elements plus the header:

cFE SB Telemetry Header (see Table 4-1)
1. Directory Name (see Table 4-2)
2. Number of Files in the Directory (see Table 4-3)
3. Number of Files in the Packet (see Table 4-4)
4. Index Into Directory Files of First Packet File (see Table 4-5)
5. Directory Listing File Data (see Table 4-6)

Table 4-1. cFE SB Telemetry Header

Name	[CFE_SB_TLM_HDR_SIZE]
Data Type	8 Bit Unsigned Integer

Table 4-2. Directory Name

Name	DirName
Data Type	Character String
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_DirName[OS_MAX_PATH_LEN]*

Table 4-3. Number of Files in the Directory	
Name	TotalFiles
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_TotalFiles*

Table 4-4. Number of Files in the Packet	
Name	PacketFiles
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_PktFiles*

Table 4-5. Index into Directory Files of First Packet File	
Name	FirstFile
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_DirOffset*

Table 4-6. Directory Listing File Data	
Name	FileList [FM_DIR_LIST_PKT_ENTRIES]
Data Type	Array
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_DirList[FM_DIR_LIST_PKT_ENTRIES]*

4.1.2 Get File Info Telemetry Packet

The Get File Info Telemetry Packet contains eight elements plus the header:

cFE SB Telemetry Header (see Table 4-7)
1. File Open or Closed Status (see Table 4-8)
2. Flag Indicating if CRC Computed or Not (see Table 4-9)
3. Structure Padding (see Table 4-10)
4. CRC Value if Computed (see Table 4-11)
5. Status – File Size (see Table 4-12)
6. Last Modification Time of File (see Table 4-13)
7. Mode of the File (Permissions) (see Table 4-14)
8. Name of File (see Table 4-15)

Table 4-7. cFE SB Telemetry Header

Name	[CFE_SB_TLM_HDR_SIZE]
Data Type	8 Bit Unsigned Integer

Table 4-8. File Open or Closed Status

Name	FileStatus
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_FileStatus*

Table 4-9. Flag Indicating if CRC Computed or Not

Name	CRC_Computed
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_ComputeCRC*

Table 4-10. Structure Padding

Name	Spare [2]
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_InfoPad[2]*

Table 4-11. CRC Value if Computed

Name	CRC_Computed
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_CRC*

Table 4-12. Status – File Size

Name	FileSize
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_InfoFileSize*

Table 4-13. Last Modification Time of File

Name	LastModifiedTime
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_ModTime*

Table 4-14. Mode of the File (Permissions)

Name	Mode
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	

Table 4-15. Name of File

Name	Filename
Data Type	Character String
Telemetry Mnemonic(s)	*cFS FM Default Value:* *sc_$cpu_FM_InfoFileName[OS_MAX_PATH_LEN]*

4.1.3 Get Free Space Telemetry Packet

The Get Free Space Telemetry Packet contains four elements plus the header:

cFE SB Telemetry Header (see Table 4-16)
1. Array of File System Free Space Entries (see Table 4-17)
2. First 32 Bit Portion of a 64 Bit Value (see Table 4-18)
3. Second 32 Bit Portion of a 64 Bit Value (see Table 4-19)
4. File System Name (see Table 4-20)

Table 4-16. cFE SB Telemetry Header

Name	[CFE_SB_TLM_HDR_SIZE]
Data Type	8 Bit Unsigned Integer

Table 4-17. Array of File System Free Space Entries

Name	FileSys [FM_TABLE_ENTRY_COUNT]
Data Type	Array
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_????*

Table 4-18. List Entry – 1st 32 Bit Portion of a 64 Bit Value

Name	FreeSpace_A
Data Type	32 Bit Unsigned Integer

Table 4-19. List Entry – 2nd 32 Bit Portion of a 64 Bit Value

Name	FreeSpace_B
Data Type	32 Bit Unsigned Integer

Table 4-20. List Entry – File System Name

Name	Name [OS_MAX_PATH_LEN]
Data Type	Character String
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_????*

4.1.4 Get Open Files Telemetry Packet

Table 4-21. The Get Open Files Telemetry Packet contains two elements plus the header:

cFE SB Telemetry Header (see Table 4-22)

1. Number of Files Opened via cFE (see Table 4-23)
2. List of Files Opened via cFE (see Table 4-24)

Table 4-22. cFE SB Telemetry Header

Name	[CFE_SB_TLM_HDR_SIZE]
Data Type	8 Bit Unsigned Integer

Table 4-23. Number of Files Opened via cFE

Name	NumOpenFiles
Data Type	32 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_TotalOpenFiles*

Table 4-24. List of Files Opened via cFE

Name	OpenFilesList[OS_MAX_NUM_OPEN_FILES]
Telemetry Mnemonic(s)	*CFS HS Default:* *$sc_$cpu_FM_OpenFileList[OS_MAX_NUM_OPEN_FILES]*

4.1.5 Housekeeping Telemetry Packet

The Housekeeping Telemetry Packet contains 10 elements plus the header:

cFE SB Telemetry Header (see Table 4-25)

1. Application Command Counter (see Table 4-26)
2. Application Command Error Counter (see Table 4-27)
3. Placeholder (see Table 4-28)
4. Number of Open Files in the System (see Table 4-29)
5. Child Task Command Counter (see Table 4-30)
6. Child Task Command Error Counter (see Table 4-31)
7. Child Task Command Warning Counter (see Table 4-32)
8. Number of Pending Commands in Queue (see Table 4-33)
9. Command Code Currently Executing (see Table 4-34)
10. Command Code Previously Executed (see Table 4-35)

Table 4-25. cFE SB Telemetry Header

Name	[CFE_SB_TLM_HDR_SIZE]
Data Type	8 Bit Unsigned Integer

Table 4-26. Application Command Counter

Name	CommandCounter
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_CMDPC*

Table 4-27. Application Command Error Counter

Name	CommandErrCounter
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_CMDEC*

Table 4-28. Placeholder

Name	Spare
Data Type	8 Bit Unsigned Integer

Table 4-29. Number of Open Files in the System

Name	NumOpenFiles
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_NumOpen*

Table 4-30. Child Task Command Counter

Name	ChildCmdCounter
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_ChildCMDPC*

Table 4-31. Child Task Command Error Counter

Name	ChildCmdErrCounter
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_ChildCMDEC*

Table 4-32. Child Task Command Warning Counter

Name	ChildCmdWarnCounter
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_ChildWarnCtr*

Table 4-33. Number of Pending Commands in Queue

Name	ChildQueueCount
Data Type	8 Bit Unsigned Integer
Telemetry Mnemonic(s)	*CFS FM Default:* *$sc_$cpu_FM_ChildQueCnt*

Table 4-34. Command Code Currently Executing

Name	ChildCurrentCC
Data Type	8 Bit Unsigned Integer

Table 4-35. Command Code Previously Executed

Name	ChildPreviousCC
Data Type	8 Bit Unsigned Integer

4.2 *Configuration Parameters*

Configuration parameters within cFS FM are mission- and platform-specific definitions.

This section shows the cFS FM configuration parameters provided by default, rather than the mission-specific values that have been incorporated into the software at the time it was finalized and compiled.

cFS FM contains 24 configuration parameters used to configure it for each platform or for a mission as a whole:

1. FM_APP_NAME (see Table 4-36)
2. FM_APP_PIPE_DEPTH (see Table 4-37)
3. FM_APP_PIPE_NAME (see Table 4-38)
4. FM_CHILD_FILE_BLOCK_SIZE (see Table 4-39)
5. FM_CHILD_FILE_LOOP_COUNT (see Table 4-39)
6. FM_CHILD_FILE_SLEEP_MS (see Table 4-39)
7. FM_CHILD_QUEUE_DEPTH (see Table 4-40)
8. FM_CHILD_SEM_NAME (see Table 4-41)
9. FM_CHILD_STAT_SLEEP_MS (see Table 4-42)
10. FM_CHILD_STAT_SLEEP_FILECOUNT (see Table 4-42)
11. FM_CHILD_TASK_NAME (see Table 4-43)
12. FM_CHILD_TASK_PRIORITY (see Table 4-45)
13. FM_CHILD_TASK_STACK_SIZE (see Table 4-46)
14. FM_DIR_LIST_FILE_DEFNAME (see Table 4-47)
15. FM_DIR_LIST_FILE_ENTRIES (see Table 4-48)
16. FM_DIR_LIST_FILE_SUBTYPE (see Table 4-49)
17. FM_DIR_LIST_PKT_ENTRIES (see Table 4-50)
18. FM_MISSION_REV (see Table 4-51)
19. FM_TABLE_CFE_NAME (see Table 4-52)
20. FM_TABLE_DEF_DESC (see Table 4-53)
21. FM_TABLE_DEF_NAME (see Table 4-54)
22. FM_TABLE_ENTRY_COUNT (see Table 4-55)
23. FM_TABLE_FILENAME (see Table 4-56)
24. FM_TABLE_VALIDATION_ERR (see Table 4-57)

Table 4-36. File Manager Application Name

Config Param	FM_APP_NAME
Value	*cFS FM Default:* *"FM"*
Description	This parameter must match the name used at startup by the cFE Executive Services when creating the FM application. Note that application names are also an argument to certain cFE commands. For example, the application name is needed to access tables via cFE Table Services commands.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Executive Services for specific information on limits related to application names.

Table 4-37. File Manager Command Pipe Depth

Config Param	FM_APP_PIPE_DEPTH
Value	*cFS FM Default:* *10*
Description	This parameter sets the total number of packets that may queue in the FM command pipe. The limit for individual message types in the queue is controlled by the default cFE Software Bus subscription limit.
Limits	There is no enforced limit for this parameter.

Table 4-38. File Manager Command Pipe Name

Config Param	FM_APP_PIPE_NAME
Value	*cFS FM Default:* *"FM_CMD_PIPE"*
Description	This parameter is the name used at startup when creating a cFE Software Bus command pipe for the FM application.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Software Bus Services for specific information on limits related to pipe names.

Table 4-39. Child Task File I/O Control Settings

Config Param	FM_CHILD_FILE_BLOCK_SIZE FM_CHILD_FILE_LOOP_COUNT FM_CHILD_FILE_SLEEP_MS
Value	*cFS FM Default:* *FM_CHILD_FILE_BLOCK_SIZE: 2048* *FM_CHILD_FILE_LOOP_COUNT: 16* *FM_CHILD_FILE_SLEEP_MS: 20*
Description	These parameters control the amount of file data that the FM child task will process before turning over control of the Central Processing Unit (CPU) to allow other tasks time to run. • FM_CHILD_FILE_BLOCK_SIZE defines the size of each block of file data that the FM child task will read or write. This value also defines the size of the FM child task I/O buffer that exists in global memory. • FM_CHILD_FILE_LOOP_COUNT defines the number of file data blocks that may be processed before the FM child task sleeps (turning over control of the CPU). • FM_CHILD_FILE_SLEEP_MS defines the length of time (in milliseconds [MS]) before the FM child task wakes (takes back control of the CPU). Note that many platforms will limit the precision of this value. Thus the combination of these three values control CPU use by the FM child task. Using a smaller block size minimizes the amount of Random-Access Memory (RAM) used by the file I/O buffer, but at the expense of file efficiency. Adjust each of the values such that the combination is appropriate for the target platform. For example, if the block size is 2048 and the loop count is 16 and the sleep time is 20, then while processing a 1 Mbyte file there will be 32 sleep cycles of 20ms each, for a total task delay of 0.64 seconds.
Limits	• FM_CHILD_FILE_BLOCK_SIZE: The FM application limits this value to be no less than 256 bytes and no greater than 32KB. • FM_CHILD_FILE_LOOP_COUNT: The FM application limits this value to be non-zero. There is no upper limit - a very large number effectively means that the FM child task will not surrender the CPU to other lower priority tasks. • FM_CHILD_FILE_SLEEP_MS: The FM application limits this value to be no greater than 100 MS. The value zero generally means a very short task delay; refer to the target platform documentation for specifics.

Table 4-40. Child Task Command Queue Entry Count

Config Param	FM_CHILD_QUEUE_DEPTH
Value	*cFS FM Default:* *3*
Description	This parameter sets the array depth for the command arguments queue in the FM main task to FM child task handshake interface. The value sets the upper limit for the number of commands that can be waiting in the queue to be processed by the low priority FM child task. A multi-entry command queue prevents the occasional slow command from being rejected because the child task has not yet completed the previous slow command.
Limits	The FM application limits this value to be no less than 1 and no greater than 10. There must be at least one because this is the method for passing command arguments from the parent to the child task. The upper limit is arbitrary.

Table 4-41. Child Task Semaphore Name - cFE Object Name

Config Param	FM_CHILD_SEM_NAME
Value	*cFS FM Default:* *"FM_CHILD_SEM"*
Description	This parameter sets the FM child task semaphore object name. The semaphore object name is required during semaphore creation by cFE Executive Services.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Executive Services for specific information on limits related to object names.

Table 4-42. Child File Stat Sleep (by File Count)

Config Param	FM_CHILD_STAT_SLEEP_FILECOUNT
Value	*cFS FM Default:* *0*
Description	FM_CHILD_STAT_SLEEP_FILECOUNT is the number of files to process (OS_stat) before sleeping. It works in tandem with FM_CHILD_STAT_SLEEP_MS (see Table 4-43 below) to reduce CPU hogging while allowing slightly more customization to balance time the operator is waiting to get data back from a directory listing versus FM hogging the CPU with calls to OS_stat.
Limits	The default is zero unless mission needs require it to be changed.

Table 4-43. Child File Stat Sleep (by Time in MS)	
Config Param	FM_CHILD_STAT_SLEEP_MS
Value	*cFS FM Default:* *0*
Description	FM_CHILD_STAT_SLEEP_MS is the number of milliseconds to sleep each cycle. One cycle is FM_CHILD_STAT_SLEEP_FILECOUNT. FM_CHILD_STAT_SLEEP_MS and FM_CHILD_STAT_SLEEP_FILECOUNT (see Table 4-42 above) work together to achieve balance between on the one hand, the time it takes to process a directory listing command and on the other hand, the time the CPU needs for other tasks. FM uses the OS_stat call to query a file's size, date, and mode when setting up directory listings, but OS_stat is a CPU intensive call. Querying many files and/or large files when processing directory listing commands can cause FM to hog the CPU. To mitigate this, FM provides options to sleep during calls for a directory listing. FM allows configuring the time for sleep and the number of directory listings before sleeping. A large sleep cycle will not hang the CPU but it may take a long time for directory listing to complete. However, a shorter sleep cycle will speed up the directory listing commands but may cause FM to hog the CPU. *In summary:* • High SLEEP_MS means less CPU hogging by FM but a longer time to process a directory listing command • Low SLEEP_MS means more potential CPU hogging by FM but shorter time to process a directory listing command • High FILECOUNT means more potential CPU hogging by FM but a shorter time to process a directory listing command • Low FILECOUNT means less CPU hogging by FM but longer time to process a directory listing command
Limits	The default is zero unless the mission needs require them to be changed.

Table 4-44. Child Task Name - cFE Object Name	
Config Param	FM_CHILD_TASK_NAME
Value	*cFS FM Default:* *"FM_CHILD_TASK"*
Description	This parameter sets the FM child task object name. The task object name is required during child task creation by cFE Executive Services.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Executive Services for specific information on limits related to object names.

Table 4-45. Child Task Execution Priority

Config Param	FM_CHILD_TASK_PRIORITY
Value	*cFS FM Default:* *205*
Description	This parameter sets the execution priority for the FM child task. It is highly recommended that this assignment be made by someone familiar with the system requirements for tasks running on the target platform. Note: This parameter is VxWorks® specific. Not all operating systems set task priority this way. **Priority Values.** A small value has higher priority than a large value. Thus, 100 is higher priority than 150. Further, a child task must have lower priority than its parent task, because a child task that runs ahead of its parent defeats the purpose of having a child task running in the background.
Limits	Value to be no less than 1 and no greater than 255.

Table 4-46. Child Task Stack Size

Config Param	FM_CHILD_TASK_STACK_SIZE
Value	*cFS FM Default:* *20480*
Description	This parameter sets the size in bytes of the FM child task stack. It is highly recommended that this assignment be made by someone familiar with the system requirements for tasks running on the target platform.
Limits	The FM application limits this value to be no less than 2048 and no greater than 20480. These limits are purely arbitrary and may need to be modified for specific platforms.

Table 4-47. Default Directory List Output Filename

Config Param	FM_DIR_LIST_FILE_DEFNAME
Value	*cFS FM Default:* *"/ram/fm_dirlist.out"*
Description	This parameter is the default output filename used by the Get Directory List to File command handler when the output filename is not provided. The default filename is used whenever the commanded output filename is an empty string.
Limits	The FM application does not place a limit on this configuration parameter. However, the symbol must be defined and the name will be subject to the same verification tests as a commanded output filename. Set this parameter to an empty string if no default filename is desired.

Table 4-48. Maximum Directory List Output File Entries

Config Param	FM_DIR_LIST_FILE_ENTRIES
Value	*cFS FM Default:* *3000*
Description	This parameter sets the upper limit for the number of directory entries that may be written to a Directory List output file. Directory List files are variable length, based on the number of directory entries actually written to the file. There may zero entries written to the file if the directory is empty. For most environments, this definition will play no role at all, as it will be set to a number much larger than the count of files that will ever exist in any directory at one time.
Limits	The FM application limits this value to be no less than 100 and no greater than 10000.

Table 4-49. Directory List Output File Header Subtype

Config Param	FM_DIR_LIST_FILE_SUBTYPE
Value	*cFS FM Default:* *12345*
Description	This parameter sets the cFE File Header sub-type value for FM Directory List data files. The value may be used to differentiate FM Directory List files from other data files.
Limits	The FM application places no limits on this unsigned 32 bit value.

Table 4-50. Directory List Telemetry Packet Entry Count

Config Param	FM_DIR_LIST_PKT_ENTRIES
Value	*cFS FM Default:* *20*
Description	This parameter sets the number of directory entries contained in the Directory List telemetry packet. The command handler will read directory entries until reaching the index of the start entry (set via command argument). The command handler will then continue to read directory entries and populate the telemetry packet either until there are no more unread directory entries or until there is a full telemetry packet.
Limits	The FM application limits this value to be no less than 10 and no greater than 100. The number of directory entries in the telemetry packet largely determine the packet size.

Table 4-51. Mission Specific Version Number for FM Application

Config Param	FM_MISSION_REV
Value	*cFS FM Default:* *0*
Description	The application version number consists of four parts: major version number, minor version number, revision number and mission specific revision number. The mission specific revision number is defined here and the other parts are defined in "fm_version.h".
Limits	Must be defined as a numeric value that is greater than or equal to zero.

Table 4-52. Free Space Table Name - cFE Object Name

Config Param	FM_TABLE_CFE_NAME
Value	*cFS FM Default:* *"FreeSpace"*
Description	Table object name is required during table creation.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Table Services for specific information on limits related to table names.

Table 4-53. Free Space Table Description

Config Param	FM_TABLE_DEF_DESC
Value	*cFS FM Default:* *"FM File System Free Space Table"*
Description	Table files contain headers that include descriptive text. This text will be put into the file header during the table make process.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. Refer to cFE Table Services for limits related to table descriptive text.

Table 4-54. Free Space Table Name - Filename with Path

Config Param	FM_TABLE_DEF_NAME
Value	*cFS FM Default:* *"/cf/apps/fm_freespace.tbl"*
Description	Table name with path is required to load table at startup.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. If the named table does not exist or fails validation, the table load will fail.

Table 4-55. Number of Free Space Table Entries

Config Param	FM_TABLE_ENTRY_COUNT
Value	*cFS FM Default:* *8*
Description	This value defines the number of entries in both the FM file system free space table and the FM file system free space telemetry packet. Note that this value does not define the number of file systems present or supported by the CFE-OSAL; the value only defines the number of file systems for which FM may be enabled to report free space data.
Limits	FM limits this value to be not less than 1 and not greater than 32.

Table 4-56. Free Space Table Name - Filename without Path

Config Param	FM_TABLE_FILENAME
Value	*cFS FM Default:* *"fm_freespace.tbl"*
Description	Table name without path defines the output name for the table file created during the table make process.
Limits	FM requires that this name be defined, but otherwise places no limits on the definition. If the table name is not valid then the make process may fail, or the table file may be unloadable to the target hardware.

Table 4-57. Table Data Validation Error Code	
Config Param	FM_TABLE_VALIDATION_ERR
Value	*cFS FM Default:* *(0xCF000080L)*
Description	Table data is verified during the table load process. Should the validation process fail, the value of FM_TABLE_VALIDATION_ERR will be returned by FM to cFE Table Services and displayed in an event message.
Limits	FM requires that this value be defined, but otherwise places no limits on the definition. Refer to cFE Table Services for limits related to error return values.

4.3 *Commands*

cFS FM has 19 commands available to ground controllers, plus one* not available to ground controllers:

1. Command 0 - No Operation (see Table 4-58)
2. Command 1 - Reset Counters (see Table 4-59)
3. Command 2 - Copy File (see Table 4-60)
4. Command 3 - Move File (see Table 4-61)
5. Command 4 - Rename File (see Table 4-62)
6. Command 5 - Delete File (see Table 4-63)
7. Command 7 - Delete All Files (see Table 4-64)
8. Command 8 - Decompress File (see Table 4-65)
9. Command 9 - Concatenate Files (see Table 4-66)
10. Command 10 - Get File Information (see Table 4-67)
11. Command 11 - Get Open Files Listing (see Table 4-68)
12. Command 12 - Create Directory (see Table 4-69)
13. Command 13 - Remove Directory (see Table 4-70)
14. Command 14 - Get Directory Listing to a File (see Table 4-71)
15. Command 15 - Get Directory Listing to a Packet (see Table 4-72)
16. Command 16 - Get Free Space (see Table 4-73)
17. Command 17 - Set Free Space Table Entry State (see Table 4-74)

   *Command 18 - Delete File - internal (see Table 4-75)

18. Command 19 - Set File Permissions (see Table 4-76)

Table 4-58. Command 0 - No Operation

Internal Name	FM_NOOP_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_NOOP*
Description	This command performs no operation except to generate an informational event that includes software version data. The command is most often used as a general aliveness test by demonstrating that the application can receive commands and generate telemetry.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Informational event FM_NOOP_CMD_EID (see Table 4-160) will be sent. • Command success counter FM_CMDPC (see Table 4-26) will increment.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length
Command Failure Verification	Failure of command may be verified with the following telemetry: • Command error counter FM_CMDEC (see Table 4-27) will increment. • Error event FM_NOOP_PKT_ERR_EID (see Table 4-87) will be sent.
Criticality	There are no critical issues related to this command.

Table 4-59. Command 1 - Reset Counters

Internal Name	FM_RESET_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_ResetCtrs*
Description	This command resets the following housekeeping telemetry: • FM_CMDPC - Command success counter(see Table 4-26) • FM_CMDEC - Command error counter (see Table 4-27) • FM_ChildCMDPC - Child command success counter (see Table 4-30) • FM_ChildCMDEC - Child command error counter (see Table 4-31) • FM_ChildWarnCtr - Child command error counter (see Table 4-32)
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command counters will be set to zero (see Description above). • Debug event FM_RESET_CMD_EID (see Table 4-168) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length
Command Failure Verification	Failure of command may be verified with the following telemetry: • Command error counter FM_CMDEC (see Table 4-27)will increment. • Error event FM_RESET_PKT_ERR_EID (see Table 4-88) will be sent.
Criticality	There are no critical issues related to this command.

Table 4-60. Command 2 - Copy File

Internal Name	FM_COPY_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_FileCopy*
Description	This command copies the source file to the target file. • The source must be an existing file and the target must not be a directory name. • If the Overwrite command argument is TRUE, then the target may be an existing file, if that file is closed. If the Overwrite command argument is FALSE, then the target must not exist. The source and target may be on different file systems. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the file will be performed by a lower priority child task. • The command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_COPY_CMD_EID (see Table 4-169) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Overwrite is not TRUE (one) or FALSE (zero) • Source filename is invalid • Source file does not exist • Source filename is a directory • Target filename is invalid • Target file already exists • Target filename is a directory • Child task interface queue is full • Child task interface logic is broken • OS copy function fails
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27), or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_COPY_PKT_ERR_EID (see Table 4-89) may be sent. • Error event FM_COPY_OVR_ERR_EID (see Table 4-90) may be sent. • Error event FM_COPY_SRC_ERR_EID (see Table 4-91) may be sent. • Error event FM_COPY_TGT_ERR_EID (see Table 4-92) may be sent. • Error event FM_COPY_CHILD_ERR_EID (see Table 4-93) may be sent. • Error event FM_COPY_OS_ERR_EID (see Table 4-94) may be sent.
Criticality	Copying files may consume file space needed by other critical tasks. In addition, copying very large files may consume more CPU resources than anticipated.
See also	Command 3 - Move File (see Table 4-61) Command 4 - Rename File (see Table 4-62)

Table 4-61. Command 3 - Move File

Internal Name	FM_MOVE_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_FileMove*

Description	This command moves the source file to the target file. • The source must be an existing file and the target must not be a directory name. • If the Overwrite command argument is TRUE, then the target may be an existing file, if the file is closed. • If the Overwrite command argument is FALSE, then the target must not exist. • Source and target must both be on the same file system. • The move command does not actually move any file data. The command modifies the file system directory structure to create a different file entry for the same file data. To move a file across file systems, first copy the file and then delete the original. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but moving the file will be performed by a lower priority child task. • The command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_MOVE_CMD_EID (see Table 4-170) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Command packet length is invalid • Overwrite is not TRUE (one) or FALSE (zero) • Source filename is invalid • Source file does not exist • Source filename is a directory • Target filename is invalid • Target file already exists • Target filename is a directory • OS move function fails • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_MOVE_PKT_ERR_EID (see Table 4-95) may be sent. • Error event FM_MOVE_OVR_ERR_EID (see Table 4-96) may be sent. • Error event FM_MOVE_SRC_ERR_EID (see Table 4-97) may be sent. • Error event FM_MOVE_TGT_ERR_EID (see Table 4-98) may be sent. • Error event FM_MOVE_OS_ERR_EID (see Table 4-100) may be sent.

Criticality	There are no critical issues related to this command.
See also	Command 2 - Copy File (see Table 4-60) Command 4 - Rename File (see Table 4-62)

Table 4-62. Command 4 - Rename File

Internal Name	FM_RENAME_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_FileRename*
Description	This command renames the source file to the target file. • Source must be an existing file and target must not exist. • Source and target must both be on the same file system. • The rename command does not actually move any file data. • The command modifies the file system directory structure to create a different file entry for the same file data. • To rename a file across file systems, first copy the file and then delete the original. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but renaming the file will be performed by a lower priority child task. • The command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Debug event FM_RENAME_CMD_EID (see Table 4-171) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Source filename is invalid • Source file does not exist • Source filename is a directory • Target filename is invalid • Target file already exists • Target filename is a directory • Failure of OS rename function • Child task interface queue is full • Child task interface logic is broken

Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) may increment. • Error event FM_RENAME_PKT_ERR_EID (see Table 4-101) may be sent. • Error event FM_RENAME_SRC_ERR_EID (see Table 4-102) may be sent. • Error event FM_RENAME_TGT_ERR_EID (see Table 4-103) may be sent. • Error event FM_RENAME_OS_ERR_EID (see Table 4-105) may be sent.
Criticality	There are no critical issues related to this command.
See also	Command 2 - Copy File (see Table 4-60) Command 3 - Move File (see Table 4-61)

Table 4-63. Command 5 - Delete File

Internal Name	FM_DELETE_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_Delete*
Description	This command deletes the source file. • Source must be an existing file that is not open. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but deleting the file will be performed by a lower priority child task. • The command result for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_DELETE_CMD_EID (see Table 4-172) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Filename is invalid • File does not exist • File is open • Filename is a directory • Failure of OS delete function • Child task interface queue is full • Child task interface logic is broken

Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_DELETE_PKT_ERR_EID (see Table 4-106) may be sent. • Error event FM_DELETE_SRC_ERR_EID (see Table 4-107) may be sent. • Error event FM_DELETE_OS_ERR_EID (see Table 4-109) may be sent.
Criticality	The FM application does not provide a method to restore deleted files. Critical data may be lost when deleting files.
See also	Command 7 - Delete All Files (see Table 4-64 Command 13 - Remove Directory (see Table 4-70)

Table 4-64. Command 7 - Delete All Files

Internal Name	FM_DELETE_ALL_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_DeleteAll*
Description	This command deletes all files in the source directory. • Source must be an existing directory. • Open files and sub-directories are not deleted. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory and deleting each file will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_DELETE_ALL_CMD_EID (see Table 4-173) will be sent.

Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Invalid directory name • Directory does not exist • Directory name + separator + filename is too long • Failure of OS delete function • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_DELETE_ALL_PKT_ERR_EID (see Table 4-110) may be sent. • Error event FM_DELETE_ALL_SRC_ERR_EID (see Table 4-111) may be sent. • Error event FM_DELETE_ALL_CHILD_ERR_EID (see Table 4-112) may be sent. • Error event FM_DELETE_ALL_OS_ERR_EID (see Table 4-113) may be sent.
Command Warning Conditions	• Directory entry is not a file (sub-directory) • Directory entry is an open file
Command Warning Verification	• Warning counter FM_ChildWarnCtr (see Table 4-32) will increment. • Informational event FM_DELETE_ALL_WARNING_EID (see Table 4-161) will be sent.
Criticality	The FM application does not provide a method to restore deleted files. Critical data may be lost when deleting files. In addition, deleting a very large number of files may consume more CPU resources than anticipated.
See also	Command 5 - Delete File (see Table 4-63) Command 13 - Remove Directory (see Table 4-70)

Table 4-65. Command 8 - Decompress File

Internal Name	FM_DECOMPRESS_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_Decompress*

Description	This command invokes a cFE function to decompress the source file into the target file. • Source must be an existing file and target must not exist. • Source and target may be on different file systems. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but decompressing the source file into the target file will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_DECOM_CMD_EID (see Table 4-174) will be sent
Command Error Conditions	This command may fail for the following reason(s): • Command packet length is invalid • Source filename is invalid • Source file does not exist • Target filename is invalid • Target file does exist • CFE_FS_Decompress (see Section 2.3.1) function fails • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_DECOM_PKT_ERR_EID (see Table 4-114) may be sent. • Error event FM_DECOM_SRC_ERR_EID (see Table 4-115) may be sent. • Error event FM_DECOM_TGT_ERR_EID (see Table 4-116) may be sent. • Error event FM_DECOM_CFE_ERR_EID (see Table 4-118) may be sent.
Criticality	Decompressing a very large file may consume more CPU resource than anticipated.

Table 4-66. Command 9 - Concatenate Files

Internal Name	FM_CONCAT_CC

Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_FileCat*
Description	This command concatenates two source files into the target file. • Sources must both be existing files and target must not exist. • Sources and target may be on different file systems. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but copying the first source file to the target file and then appending the second source file to the target file will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_CONCAT_CMD_EID (Table 4-175) will be sent
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Invalid source filename • Source file does not exist • Invalid target filename • Target file does exist • Failure of OS function (copy, open, read, write, etc.) • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_CONCAT_PKT_ERR_EID (see Table 4-119) may be sent. • Error event FM_CONCAT_SRC1_ERR_EID (see Table 4-120) may be sent. • Error event FM_CONCAT_SRC2_ERR_EID (see Table 4-121) may be sent. • Error event FM_CONCAT_TGT_ERR_EID (see Table 4-122) may be sent. • Error event FM_CONCAT_OS_ERR_EID (see Table 4-124) may be sent
Criticality	Concatenating very large files may consume more CPU resource than anticipated.
See also:	Command 2 - Copy File (see Table 4-60)

Table 4-67. Command 10 - Get File Information	
Internal Name	FM_GET_FILE_INFO_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_FileInfo*
Description	This command creates an FM file information telemetry packet for the source file. • The file information packet includes status that indicates whether source is a file that is open or closed, a directory, or does not exist. • The file information data also includes a Cyclic Redundancy Check (CRC), file size, last modify time, and the source name. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but collecting the status data and calculating the CRC will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_GET_FILE_INFO_CMD_EID (see Table 4-176) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Invalid source filename • Failure of OS_stat function • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_GET_FILE_INFO_PKT_ERR_EID (see Table 4-125) may be sent. • Error event FM_GET_FILE_INFO_SRC_ERR_EID (see Table 4-126) may be sent. • Error event FM_GET_FILE_INFO_CHILD_ERR_EID (see Table 4-127) may be sent.
Criticality	Calculating the CRC for a very large file may consume more CPU resource than anticipated.

See also	Command 11 - Get Open Files Listing (see Table 4-68 Command 14 - Get Directory Listing to a File (see Table 4-71) Command 15 - Get Directory Listing to a Packet (see Table 4-72

Table 4-68. Command 11 - Get Open Files Listing

Internal Name	FM_GET_OPEN_FILES_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_ListOpenFiles*
Description	This command creates an FM open files telemetry packet. The open files packet includes the number of open files and for each open file, the name of the file and the name of the application that opened the file.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Debug event FM_GET_OPEN_FILES_CMD_EID (see Table 4-177) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length
Command Failure Verification	Failure of command may be verified with the following telemetry: • Command error counter FM_CMDEC (see Table 4-27) will increment. • Error event FM_GET_OPEN_FILES_PKT_ERR_EID will be sent.
Criticality	There are no critical issues related to this command.
See also	Command 10 - Get File Information (see Table 4-67) Command 14 - Get Directory Listing to a File (see Table 4-71) Command 15 - Get Directory Listing to a Packet (see Table 4-72)

Table 4-69. Command 12 - Create Directory

Internal Name	FM_CREATE_DIR_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_DirCreate*
Description	This command creates the source directory. • Source must be a valid directory name that does not exist. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but creating the directory will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.

Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_CREATE_DIR_CMD_EID (see Table 4-178) will be sent. • Command 14 - Get Directory Listing to a File (see Table 4-71) or Command 15 - Get Directory Listing to a Packet (see Table 4-72) can be issued to verify further that the directory has been created.
Command Error Conditions	This command may fail for the following reason(s): • Invalid command packet length • Invalid directory name • Directory name already exists • OS_mkdir function fails • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_CREATE_DIR_PKT_ERR_EID (see Table 4-129) may be sent. • Error event FM_CREATE_DIR_SRC_ERR_EID (see Table 4-130) may be sent. • Error event FM_CREATE_DIR_OS_ERR_EID (see Table 4-132) may be sent.
Criticality	There are no critical issues related to this command.
See also	Command 13 - Remove Directory (see Table 4-70)

Table 4-70. Command 13 - Remove Directory

Internal Name	FM_DELETE_DIR_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_DirDelete*

Description	This command deletes the source directory, but not the contents of the directory. • Source must be a valid existing directory name. • Source directory must be empty. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but deleting the directory will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_DELETE_DIR_CMD_EID (see Table 4-179) will be sent. • Command 14 - Get Directory Listing to a File (see Table 4-71) or Command 15 - Get Directory Listing to a Packet (see Table 4-72) can be issued to verify further that the directory has been deleted.
Command Error Conditions	This command may fail for the following reason(s): • Command packet length is invalid • Directory name is invalid • Directory does not exist • Directory is not empty • OS function (OS_opendir, OS_rmdir) fails • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_DELETE_DIR_PKT_ERR_EID (see Table 4-133) may be sent. • Error event FM_DELETE_DIR_SRC_ERR_EID (see Table 4-134) may be sent. • Error event FM_DELETE_DIR_EMPTY_ERR_EID (see Table 4-136) may be sent. • Error event FM_DELETE_DIR_OS_ERR_EID (see Table 4-137) may be sent.
Criticality	The unexpected loss of a directory may affect a critical tasks ability to store data.
See also	Command 12 - Create Directory (see Table 4-69)

Table 4-71. Command 14 - Get Directory Listing to a File	
Internal Name	FM_GET_DIR_FILE_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_DirListFile*
Description	This command writes a listing of the contents of the source directory to the target file. • If the target filename buffer is empty, then the default target filename FM_DIR_LIST_FILE_DEFNAME (see Table 4-47) is used. • The command will overwrite a previous copy of the target file, if one exists. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • Debug event FM_GET_DIR_FILE_CMD_EID (see Table 4-180) will be sent.
Command Error Conditions	This command may fail for the following reason(s): • Command packet length is invalid • Source directory name is invalid • Source directory does not exist • Directory name + separator is too long • Directory name + directory entry is too long • Target filename is invalid • Target file is already open • OS function (OS_opendir, OS_creat, OS_write) fails • Child task interface queue is full • Child task interface logic is broken
Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) or child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_GET_DIR_FILE_OS_ERR_EID (see Table 4-142) may be sent.
Criticality	Reading a directory that contains thousands of files may consume more CPU resource than anticipated.
See also	Command 15 - Get Directory Listing to a Packet (see Table 4-72)

Table 4-72. Command 15 - Get Directory Listing to a Packet	
Internal Name	FM_GET_DIR_PKT_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_DirListTlm*
Description	This command creates a Get Directory Listing Telemetry Packet (see Section 4.1.1) that contains a listing of the entries in the specified directory. • Since the packet will likely hold fewer entries than will be possible in a directory, the command also provides an index argument (see Table 4-5) to define which entry in the directory is the first entry reported in the telemetry packet. • After reading the directory list and skipping entries until reaching the index of the first entry reported, the remaining entries in the packet are filled sequentially until either the packet is full or until there are no more entries in the directory. • The first entry index is zero based - thus, when the first entry index is zero the first directory entry will be the first packet entry. The number of entries per packet FM_DIR_LIST_PKT_ENTRIES (see Table 4-50) is a platform configuration definition. • Because of the possibility that this command might take a very long time to complete, command argument validation will be done immediately but reading the directory will be performed by a lower priority child task. • The return value for this function only refers to the result of command argument verification and being able to place the command on the child task interface queue.
Command Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion. • The Get Directory Listing Telemetry Packet (see Section 4.1.1) will be sent. • The FM_GET_DIR_PKT_CMD_EID (see Table 4-181) debug event will be sent.
Command Error Conditions	• This command may fail for the following reason(s): • OS error received opening directory • OS error received requesting directory size • OS error received closing directory • Directory pathname received is invalid • Command packet length not as expected • Child task interface queue is full • Child task interface logic is broken

Command Failure Verification	Failure of command may be verified with the following telemetry: • Either command error counter FM_CMDEC (see Table 4-27) child command error counter FM_ChildCMDEC (see Table 4-31) will increment. • Error event FM_GET_DIR_PKT_PKT_ERR_EID (see Table 4-143) may be sent. • Error event FM_GET_DIR_PKT_SRC_ERR_EID (see Table 4-144) may be sent. • Error event FM_GET_DIR_PKT_CHILD_ERR_EID (see Table 4-145) may be sent. • Error event FM_GET_DIR_PKT_OS_ERR_EID (see Table 4-146) may be sent.
Command Warning Conditions	If directory contents would exceed packet length.
Command Warning Verification	• Warning counter FM_ChildWarnCtr (see Table 4-32) will increment. • Informational event FM_GET_DIR_PKT_WARNING_EID (see Table 4-164) will be sent.
Criticality	Reading a directory that contains thousands of files may consume more CPU resource than anticipated.
See also	Command 14 - Get Directory Listing to a File (see Table 4-71)

Table 4-73. Command 16 - Get Free Space

Internal Name	FM_GET_FREE_SPACE_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpu_FM_GetFreeSpace*
Description	This command queries the amount of free space for each of the enabled entries in the file system free space table. The data is then placed in a telemetry packet and sent on the cFE Software Bus.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Debug event FM_GET_FREE_SPACE_CMD_EID (see Table 4-182) will be sent. • Get Free Space Telemetry Packet (see Section 4.1.3) will be sent.
Error Conditions	This command may fail for the following reasons: • Command packet length is invalid • FM file system free space table has not yet been loaded

Command Failure Verification	Failure of command may be verified with the following telemetry: • Command error counter FM_CMDEC (see Table 4-27) will increment. • Error event FM_GET_FREE_SPACE_PKT_ERR_EID (see Table 4-147) may be sent. • Error event FM_GET_FREE_SPACE_TBL_ERR_EID (see Table 4-148) may be sent.
Criticality	There are no critical issues related to this command.

Table 4-74. Command 17 - Set Free Space Table Entry State

Internal Name	FM_SET_TABLE_STATE_CC
Command Mnemonic(s)	*cFS FM Default:* *$sc_$cpuFM_GetTblState*
Description	This command enables or disables a single entry in the FM file system free space table. • Only table entries that are currently enabled or disabled may be modified by command. • Unused table entries cannot be modified.
Command Success Verification	Successful execution of this command may be verified with the following telemetry: • Command success counter FM_CMDPC (see Table 4-26) will increment. • Informational event FM_SET_TABLE_STATE_CMD_EID (see Table 4-165) will be sent.
Error Conditions	This command may fail for the following reason(s): • Command packet length is invalid • FM file system free space table has not yet been loaded • Invalid command argument, table entry index arg is out of range • Command argument invalid; entry state argument is not enable/disable • Current table entry state invalid; entry is unused
Command Failure Verification	Failure of command may be verified with the following telemetry: • Command error counter FM_CMDEC (see Table 4-27) will increment. • Error event FM_SET_TABLE_STATE_PKT_ERR_EID (see Table 4-149) may be sent. • Error event FM_SET_TABLE_STATE_TBL_ERR_EID (see Table 4-150) may be sent. • Error event FM_SET_TABLE_STATE_ARG_ERR_EID (see Table 4-151) may be sent. • Error event FM_SET_TABLE_STATE_UNUSED_ERR_EID (see Table 4-152) may be sent.
Criticality	There are no critical issues related to this command.

Table 4-75. Command 18 - Delete File (internal)

Internal Name	FM_DELETE_INT_CC

Command No.	18
Description	This is a special version of the FM_DELETE_CC command. This command message is intended to be used only within the spacecraft. It is not intended to be sent from the ground.
See also	Refer to FM_DELETE_CC command (see Table 4-63) for use details.
Command Success Verification	This version of the command will not generate a success event, nor will the command increment the command success counter.

Table 4-76. Command 19 - Set File Permissions

Command Name	FM_SET_FILE_PERM_CC
Command Mnemonic(s)	*cFS FM Default:* *FM_SetFilePerm*
Description	This command sets the permissions for a file. This is a direct interface to OS_chmod in the OSAL. OS_chmod accepts a 32 bit unsigned integer to set the file's mode. The mode value also contains the type of file (regular or directory, etc.) so care should be taken to not change the file type from regular to directory or vice-versa. Examples for a regular file: • 0100700 (Decimal: 33216) - Read, Write and Execute • 0100600 (Decimal: 33152) - Read, and Write • 0100400 (Decimal: 33024) - Read Only Examples for a directory: • 0040700 (Decimal: 16832) - Read, Write and Execute • 0040600 (Decimal: 16786) - Read, and Write • 0040400 (Decimal: 16640) - Read Only Other examples: • S_IFMT 0170000 bit mask for the file type bit field • S_IFSOCK 0140000 socket • S_IFLNK 0120000 symbolic link • S_IFREG 0100000 regular file • S_IFBLK 0060000 block device • S_IFDIR 0040000 directory • S_IFCHR 0020000 character device • S_IFIFO 0010000 FIFO

Command Success Verification	• Command success counter FM_CMDPC (see Table 4-26) will increment after validation. • Child command success counter FM_ChildCMDPC (see Table 4-30) will increment after completion.
Error Conditions	• Invalid command packet length • Error from call to OS_chmod
Command Failure Verification	Failure of command may be verified with the following telemetry: • Command error counter FM_CMDEC (see Table 4-27) will increment • Error event FM_SET_PERM_ERR_EID (see Table 4-157) may be sent • Error event FM_SET_PERM_CMD_EID (see Table 4-183) may be sent • Error event FM_SET_PERM_OS_ERR_EID (see Table 4-158) may be sent
Criticality	There are no critical issues related to this command.

4.4 ***Event Messages***

This section shows event messages. Event messages are informational text generated by the application in response to commands, software errors, hardware errors, application initialization, and other conditions. Event messages are sent to alert that a significant event on board has occurred. Event messages may also be sent for debugging application code during development, maintenance, and testing.

Event messages are not mission-specific. Generally the event messages shown below should appear in telemetry on ground FSW systems (e.g., ASIST or ITOS), no matter the mission.

The levels are not used for programmatic control by cFS FM. In other words, cFS FM does not treat the labels differently. However, ASIST, ITOS, or other ground flight control software may differentiate them.

There are four levels of event types:

- Level 1 - Critical (see Section 4.4.1)
- Level 2 - Error (see Section 4.4.2)
- Level 3 - Information (see Section 4.4.3)
- Level 4 - Debug (see Section 4.4.4)

4.4.1 Event Messages - CRITICAL

The tables in this section show **critical** event messages for cFS FM. Events of this type are notifications of error conditions that cFS FM is unable to correct or compensate for. These might be uncorrectable memory errors, hardware failures, etc.

No critical event messages are included in cFS FM at this time.

4.4.2 Event Messages - ERROR

For an index of all event messages, see Appendix A .

The tables in this section show **error** event messages for cFS FM. Events of this type are notifications of abnormal behavior. However, they represent error conditions that have been identified and corrected for by the flight software. These typically represent things like erroneous commands, illegal mode change attempts, switching to redundant hardware, etc.

cFS FM contains 82 error event messages.

1. EID 2 - 'Initialization Error: Register for Event Services' (see Table 4-77)
2. EID 3 - 'Initialization Error: Create SB Input Pipe' (see Table 4-78)
3. EID 4 - 'Initialization Error: Subscribe to HK Request' (see Table 4-79)
4. EID 5 - 'Initialization Error: Subscribe to FM Commands' (see Table 4-80)
5. EID 6 - 'Initialization Error: Register Free Space Table' (see Table 4-81)
6. EID 7 - 'Main Loop Error: Software Bus Receive' (see Table 4-82)
7. EID 8 - 'Application Terminating' (see Table 4-83)
8. EID 9 - 'Main Loop Error: Invalid Message ID' (see Table 4-84)
9. EID 10 - 'Main Loop Error: Invalid Command Code' (see Table 4-85)
10. EID 11 - 'HK Request Error: Invalid Command Packet Length' (see Table 4-86)
11. EID 13 - 'No-op Error: Invalid Command Packet Length' (see Table 4-87)
12. EID 15 - 'Reset Counters Error: Invalid Command Packet Length' (see Table 4-88)
13. EID 17 - 'Copy File Error: Invalid Command Packet Length' (see Table 4-89)
14. EID 18 - 'Copy File Error: Invalid Overwrite' (see Table 4-90)
15. EID 19 - 'Copy File Error: Source Filename' (see Table 4-91)
16. EID 20 - 'Copy File Error: Target Filename' (see Table 4-92)
17. EID 21 - 'Copy File Error: Child Task' (see Table 4-93)

18. EID 22 - 'Copy File Error: OS Error' (see Table 4-94)
19. EID 24 - 'Move File Error: Invalid Command Packet Length' (see Table 4-95)
20. EID 25 - 'Move File Error: Invalid Overwrite' (see Table 4-96)
21. EID 26 - 'Move File Error: Source Filename' (see Table 4-97)
22. EID 27 - 'Move File Error: Target Filename' (see Table 4-98)
23. EID 28 - 'Move File Error: Child Task' (see Table 4-99)
24. EID 29 - 'Move File Error: OS Error' (see Table 4-100)
25. EID 31 - 'Rename File Error: Invalid Command Packet Length' (see Table 4-101)
26. EID 32 - 'Rename File Error: Source Filename' (see Table 4-102)
27. EID 33 - 'Rename File Error: Target Filename' (see Table 4-103)
28. EID 34 - 'Rename File Error: Child Task' (see Table 4-104)
29. EID 35 - 'Rename File Error: OS Error' (see Table 4-105)
30. EID 37 - 'Delete File Error: Invalid Command Packet Length' (see Table 4-106)
31. EID 38 - 'Delete File Error: Filename' (see Table 4-107)
32. EID 39 - 'Delete File Error: Child Task' (see Table 4-108)
33. EID 40 - 'Delete File Error: OS Error' (see Table 4-109)
34. EID 43 - 'Delete All Files Error: Invalid Command Packet Length' (see Table 4-110)
35. EID 44 - 'Delete All Files Error: Directory Name' (see Table 4-111)
36. EID 45 - 'Delete All Files Error: Child Task' (see Table 4-112)
37. EID 46 - 'Delete All Files Error: OS Error' (see Table 4-113)
38. EID 48 - 'Decompress File Error: Invalid Command Packet Length' (see Table 4-114)
39. EID 49 - 'Decompress File Error: Source Filename' (see Table 4-115)
40. EID 50 - 'Decompress File Error: Target Filename' (see Table 4-116)
41. EID 51 - 'Decompress File Error: Child Task' (see Table 4-117)
42. EID 52 - 'Decompress File Error: CFE Error' (see Table 4-118)
43. EID 54 - 'Concat Files Error: Invalid Command Packet Length' (see Table 4-119)
44. EID 55 - 'Concat Files Error: Source 1 Filename' (see Table 4-120)
45. EID 56 - 'Concat Files Error: Source 2 Filename' (see Table 4-121)
46. EID 57 - 'Concat Files Error: Target Filename' (see Table 4-122)
47. EID 58 - 'Concat File Error: Child Task' (see Table 4-123)
48. EID 59 - 'Concat File Error: OS Error' (see Table 4-124)
49. EID 61 - 'Get File Info Error: Invalid Command Packet Length' (see Table 4-125)
50. EID 62 - 'Get File Info Error: Source Filename' (see Table 4-126)
51. EID 63 - 'Get File Info Error: Child Task' (see Table 4-127)
52. EID 66 - 'Get Open Files Error: Invalid Command Packet Length' (see Table 4-128)
53. EID 68 - 'Create Directory Error: Invalid Command Packet Length' (see Table 4-129)
54. EID 69 - 'Create Directory Error: Directory Name' (see Table 4-130)
55. EID 70 - 'Create Directory File Error: Child Task' (see Table 4-131)
56. EID 71 - 'Create Directory Error: OS Error' (see Table 4-132)
57. EID 73 - 'Delete Directory Error: Invalid Command Packet Length' (see Table 4-133)
58. EID 74 - 'Delete Directory Error: Directory Name' (see Table 4-134)
59. EID 75 - 'Delete Directory File Error: Child Task' (see Table 4-135)
60. EID 76 - 'Delete Directory Error: Source Filename' (see Table 4-136)
61. EID 77 - 'Delete Directory Error: OS Error' (see Table 4-137)
62. EID 79 - 'Directory List to File Error: Invalid Command Packet Length' (see Table 4-138)
63. EID 80 - 'Directory List to File Error: Directory Name' (see Table 4-139)
64. EID 81 - 'Directory List to File Error: Output Filename' (see Table 4-140)
65. EID 83 - 'Directory List to File Error: Child Task' (see Table 4-141)
66. EID 84 - 'Directory List to File Error: OS Error' (see Table 4-142)
67. EID 87 - 'Directory List to Packet Error: Invalid Command Packet Length' (see Table 4-143)
68. EID 88 - 'Directory List to Packet Error: Directory Name' (see Table 4-144)

69. EID 89 - 'Directory List to Packet Error: Child Task' (see Table 4-145)
70. EID 90 - 'Directory List to Packet Error: OS Error' (see Table 4-146)
71. EID 92 - 'Get Free Space Error: Invalid Command Packet Length' (see Table 4-147)
72. EID 93 - 'Get Free Space Error: Table Not Loaded' (see Table 4-148)
73. EID 95 - 'Set Table State Error: Invalid Command Packet Length' (see Table 4-149)
74. EID 96 - 'Set Table State Error: Table Not Loaded' (see Table 4-150)
75. EID 97 - 'Set Table State Error: Invalid Command Argument' (see Table 4-151)
76. EID 98 - 'Set Table State Error: Unused Table Entry' (see Table 4-152)
77. EID 99 - 'Free Space Table Verify Error' (see Table 4-153)
78. EID 101 - 'Child Task Initialization Error' (see Table 4-154)
79. EID 102 - 'Child Task Termination Error' (see Table 4-155)
80. EID 103 - 'Child Task Execution Error' (see Table 4-156)
81. EID 110 - 'Set Permissions error: Invalid Command Packet Length' (see Table 4-157)
82. EID 112 - 'Set Permissions: OS_chmod error' (see Table 4-158)

Table 4-77. (ERROR) EID 2 - 'Initialization Error: Register for Event Services'

Internal Name	FM_STARTUP_EVENTS_ERR_EID
Event Message	'Initialization error: register for event services: result = [result]' *Where:* • [result] is the error code returned from the call to the API function CFE_EVS_Register.
Cause	This event message is issued when the File Manager application has failed in its attempt to register for event services during startup initialization. This is a fatal error. It will cause the File Manager application to terminate.

Table 4-78. (ERROR) EID 3 - 'Initialization Error: Create SB Input Pipe'

Internal Name	FM_STARTUP_CREAT_PIPE_ERR_EID
Event Message	'Initialization error: create SB input pipe: result = [result]' *Where:* • [result] is the error code returned from the call to the API function CFE_SB_CreatePipe.
Cause	This event message is issued when the File Manager application has failed in its attempt to create a Software Bus input pipe during startup initialization. This is a fatal error. It will cause the File Manager application to terminate.

Table 4-79. (ERROR) EID 4 - 'Initialization Error: Subscribe to HK Request'

Internal Name	FM_STARTUP_SUBSCRIB_HK_ERR_EID
Event Message	'Initialization error: subscribe to HK request: result = [result]' *Where:* • [result] is the error code returned from the call to the API function CFE_SB_Subscribe.
Cause	This event message is issued when the File Manager application has failed in its attempt to subscribe to the HK telemetry request command during startup initialization. This is a fatal error. It will cause the File Manager application to terminate.

Table 4-80. (ERROR) EID 5 - 'Initialization Error: Subscribe to FM Commands'

Internal Name	FM_STARTUP_SUBSCRIB_GCMD_ERR_EID
Event Message	'Initialization error: subscribe to FM commands: result = [result]' *Where:* • [result] is the error code returned from the call to the API function CFE_SB_Subscribe.
Cause	This event message is issued when the File Manager application has failed in its attempt to subscribe to the FM ground command packet during startup initialization. This is a fatal error. It will cause the File Manager application to terminate.

Table 4-81. (ERROR) EID 6 - 'Initialization Error: Register Free Space Table'

Internal Name	FM_STARTUP_TABLE_INIT_ERR_EID
Event Message	'Initialization error: register free space table: result = [result]' *Where:* • [result] is the error code returned from the call to the API function CFE_TBL_Register.
Type	Error
Cause	This event message is issued when the File Manager application has failed in its attempt to register its file system free space table during startup initialization. This is a fatal error. It will cause the File Manager application to terminate.

Table 4-82. (ERROR) EID 7 - 'Main Loop Error: Software Bus Receive'

Internal Name	FM_SB_RECEIVE_ERR_EID
Event Message	'Main Loop Error: SB receive: result = [result]' *Where:* • [result] is the error code returned from the call to the API function CFE_SB_RcvMsg.
Cause	This event message is issued when the File Manager application has failed in its attempt to read from its Software Bus input pipe while processing the software main loop sequence. This is a fatal error. It will cause the File Manager application to terminate.

Table 4-83. (ERROR) EID 8 - 'Application Terminating'

Internal Name	FM_EXIT_ERR_EID
Event Message	'Application terminating: result = [result]' *Where:* • [result] is the nonzero error code from a fatal error that has occurred.
Cause	This event message is issued when the File Manager application is about to terminate. Fatal errors all have descriptive events. If the result value in the event text is zero, then it is likely that the cFE has terminated the FM application, presumably by command.

Table 4-84. (ERROR) EID 9 - 'Main Loop Error: Invalid Message ID'

Internal Name	FM_MID_ERR_EID
Event Message	'Main loop error: invalid message ID: mid = [mid]' *Where:* • [mid] is the unexpected MessageID.
Cause	This event message is issued when the File Manager application has received an unexpected Software Bus packet. There is no obvious explanation of why or how FM could receive such a packet.

Table 4-85. (ERROR) EID 10 - 'Main Loop Error: Invalid Command Code'

Internal Name	FM_CC_ERR_EID
Event Message	'Main loop error: invalid command code: cc = [CC]' *Where:* • [CC] is the unexpected command code.
Cause	This event message is issued when the File Manager application has received a command packet with an unexpected command code value. Malformed command packets are generally prevented by the ground system. Therefore, the source for the problem command is likely to be one of the on-board tables that contain commands.

Table 4-86. (ERROR) EID 11 - 'HK Request Error: Invalid Command Packet Length'

Internal Name	FM_HK_REQ_ERR_EID
Event Message	'HK Request error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a housekeeping request command packet with an invalid length.

Table 4-87. (ERROR) EID 13 - 'Noop Error: Invalid Command Packet Length'

Internal Name	FM_NOOP_PKT_ERR_EID
Event Message	'Noop error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 0 - No Operation telemetry packet (see Table 4-58) with an invalid length.

Table 4-88. (ERROR) EID 15 - 'Reset Counters Error: Invalid Command Packet Length'

Internal Name	FM_RESET_PKT_ERR_EID
Event Message	'Reset Counters error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 1 - Reset Counters (see Table 4-59) telemetry packet with an invalid length.

Table 4-89. (ERROR) EID 17 - 'Copy File Error: Invalid Command Packet Length'

Internal Name	FM_COPY_PKT_ERR_EID
Event Message	'Copy File error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 2 - Copy File (see Table 4-60) telemetry packet with an invalid length.

Table 4-90. (ERROR) EID 18 - 'Copy File Error: Invalid Overwrite'

Internal Name	FM_COPY_OVR_ERR_EID
Event Message	'Copy File error: invalid overwrite = [invalidoverwrite]' *Where:* • [invalidoverwrite] is the invalid numeric overwrite argument
Cause	This event message is generated upon receipt of a Command 2 - Copy File (see Table 4-60) telemetry packet with an invalid overwrite argument. Overwrite must be set to TRUE (one) or FALSE (zero).

Table 4-91. (ERROR) EID 19 - 'Copy File Error: Source Filename'

Internal Name	FM_COPY_SRC_ERR_EID
Event Message	'Copy File error: filename is invalid: name = [name]' 'Copy File error: file does not exist: name = [name]' 'Copy File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid source filename, • a pre-existing source file, or • a pre-existing directory with the same name as the specified target file
Cause	This event message is generated upon receipt of a Command 2 - Copy File (see Table 4-60) telemetry packet with an invalid source filename.

Table 4-92. (ERROR) EID 20 - 'Copy File Error: Target Filename'

Internal Name	FM_COPY_TGT_ERR_EID
Event Message	'Copy File error: filename is invalid: name = [name]' 'Copy File error: file already exists: name = [name]' 'Copy File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid target filename, • a pre-existing target file, or • a pre-existing directory with the same name as the specified target file
Cause	This event message is generated upon receipt of a Command 2 - Copy File (see Table 4-60) telemetry packet with an invalid target filename.

Table 4-93. (ERROR) EID 21 - 'Copy File Error: Child Task'	
Internal Name	FM_COPY_CHILD_ERR_EID
Event Message	'Copy File error: child task is disabled' 'Copy File error: child task queue is full' 'Copy File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-94. (ERROR) EID 22 - 'Copy File Error: OS Error'	
Internal Name	FM_COPY_OS_ERR_EID
Event Message	'Copy File error: OS_cp failed: result = [result], src = [src], tgt = [tgt]' *Where:* • [result] is the OS specific return value from the function call. • [src] is the source name of the file being copied. • [tgt] is the target filename.
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the source file exists and the target name is unused and appears to be valid. • Verify that the target filename is reasonable. • Verify that the file system has sufficient free space for this operation. • Refer to the OS specific return value.

Table 4-95. (ERROR) EID 24 - 'Move File Error: Invalid Command Packet Length'	
Internal Name	FM_MOVE_PKT_ERR_EID
Event Message	'Move File error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 3 - Move File (see Table 4-61) telemetry packet with an invalid length.

Table 4-96. (ERROR) EID 25 - 'Move File Error: Invalid Overwrite'	
Internal Name	FM_MOVE_OVR_ERR_EID
Event Message	'Move File error: invalid overwrite = [overwrite]' *Where:* • [overwrite] is the invalid overwrite argument, which must be set to TRUE (one) or FALSE (zero)
Cause	This event message is generated upon receipt of a Command 3 - Move File (see Table 4-61) telemetry packet with an invalid overwrite argument.

Table 4-97. (ERROR) EID 26 - 'Move File Error: Source Filename'	
Internal Name	FM_MOVE_SRC_ERR_EID
Event Message	'Move File error: filename is invalid: name = [name]' 'Move File error: file does not exist: name = [name]' 'Move File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid source filename, • a nonexistent source file, or • a pre-existing directory with the same name as the specified source file
Cause	This event message is generated upon receipt of a Command 3 - Move File (see Table 4-61) telemetry packet with an invalid source filename.

Table 4-98. (ERROR) EID 27 - 'Move File Error: Target Filename'	
Internal Name	FM_MOVE_TGT_ERR_EID
Event Message	'Move File error: filename is invalid: name = [name]' 'Move File error: file already exists: name = [name]' 'Move File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid target filename, • a pre-existing target file, or • a pre-existing directory with the same name as the specified target file
Cause	This event message is generated upon receipt of a Command 3 - Move File (see Table 4-61) telemetry packet with an invalid target filename.

Table 4-99. (ERROR) EID 28 - 'Move File Error: Child Task'	
Internal Name	FM_MOVE_CHILD_ERR_EID
Event Message	'Move File error: child task is disabled' 'Move File error: child task queue is full' 'Move File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-100. (ERROR) EID 29 - 'Move File Error: OS Error'	
Internal Name	FM_MOVE_OS_ERR_EID
Event Message	'Move File error: OS_mv error = [error], src = [src], tgt = [tgt]' *Where:* • [error] is the return value from the function call • [src] is the source name for the file being moved. • [tgt] is the target name for the file being moved.
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the source file exists and the target name is unused and appears to be valid. Verify that the target filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return value.

Table 4-101. (ERROR) EID 31 - 'Rename File Error: Invalid Command Packet Length'	
Internal Name	FM_RENAME_PKT_ERR_EID
Event Message	'Rename File error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 4 - Rename File (see Table 4-62) telemetry packet with an invalid length.

Table 4-102. (ERROR) EID 32 - 'Rename File Error: Source Filename'

Internal Name	FM_RENAME_SRC_ERR_EID
Event Message	'Rename File error: filename is invalid: name = [name]' 'Rename File error: file does not exist: name = [name]' 'Rename File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid source filename, • a nonexistent source file, or • a pre-existing directory with the same name as the specified source file
Cause	This event message is generated upon receipt of a Command 4 - Rename File (see Table 4-62) telemetry packet with an invalid source filename.

Table 4-103. (ERROR) EID 33 - 'Rename File Error: Target Filename'

Internal Name	FM_RENAME_TGT_ERR_EID
Event Message	'Rename File error: filename is invalid: name = [name]' 'Rename File error: file already exists: name = [name]' 'Rename File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid target filename, • a pre-existing target file, or • a pre-existing directory with the same name as the specified target file
Cause	This event message is generated upon receipt of a Command 4 - Rename File (see Table 4-62) telemetry packet with an invalid target filename.

Table 4-104. (ERROR) EID 34 - 'Rename File Error: Child Task'

Internal Name	FM_RENAME_CHILD_ERR_EID
Event Message	'Rename File error: child task is disabled' 'Rename File error: child task queue is full' 'Rename File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-105. (ERROR) EID 35 - 'Rename File Error: OS Error'

Internal Name	FM_RENAME_OS_ERR_EID
Event Message	'Rename File error: OS_rename error = [error], src = [src], tgt = [tgt]' *Where:* • [error] is the return value from the OS function call • [src] is the source name for the file being renamed • [tgt] is the target name for the file being renamed
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the source file exists and the target name is unused and appears to be valid. Verify that the target filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return value.

Table 4-106. (ERROR) EID 37 - 'Delete File Error: Invalid Command Packet Length'

Internal Name	FM_DELETE_PKT_ERR_EID
Event Message	'Delete File error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 5 - Delete File (see Table 4-63) telemetry packet with an invalid length.

Table 4-107. (ERROR) EID 38 - 'Delete File Error: Filename'

Internal Name	FM_DELETE_SRC_ERR_EID
Event Message	'Delete File error: filename is invalid: name = [name]' 'Delete File error: file does not exist: name = [name]' 'Delete File error: file is already open: name = [name]' 'Delete File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid source filename, • a nonexistent source file, • an open file, or • a pre-existing directory with the same name as the specified source file
Cause	This event message is generated upon receipt of a Command 5 - Delete File (see Table 4-63) telemetry packet with an invalid filename.

Table 4-108. (ERROR) EID 39 - 'Delete File Error: Child Task'

Internal Name	FM_DELETE_CHILD_ERR_EID
Event Message	'Delete File error: child task is disabled' 'Delete File error: child task queue is full' 'Delete File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-109. (ERROR) EID 40 - 'Delete File Error: OS Error'

Internal Name	FM_DELETE_OS_ERR_EID
Event Message	'Delete File error: OS_remove error = [error], file = [file]' *Where:* • [error] is the return value from the OS function call • [file] is the file to be deleted.
Cause	This event message is generated due to an OS function error that occurs after preliminary command argument verification tests indicate that the filename exists and is not open. Refer to the OS-specific return value for an indication of what might have caused this error.

Table 4-110. (ERROR) EID 43 - 'Delete All Files Error: Invalid Command Packet Length'

Internal Name	FM_DELETE_ALL_PKT_ERR_EID
Event Message	'Delete All Files error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected packet length • [actual] is the actual packet length
Cause	This event message is generated upon receipt of a Command 7 - Delete All Files (see Table 4-64) telemetry packet with an invalid length.

Table 4-111. (ERROR) EID 44 - 'Delete All Files Error: Directory Name'

Internal Name	FM_DELETE_ALL_SRC_ERR_EID
Event Message	'Delete All Files error: directory name is invalid: name = [name]' 'Delete All Files error: directory does not exist: name = [name]' 'Delete All Files error: directory name exists as a file: name = [name]' *Where [name] is, respectively:* • the invalid directory name, • a nonexistent directory, or • a pre-existing file with the same name as the specified directory
Cause	This event message is generated upon receipt of a Command 7 - Delete All Files (see Table 4-64) telemetry packet with an invalid directory name.

Table 4-112. (ERROR) EID 45 - 'Delete All Files Error: Child Task'

Internal Name	FM_DELETE_ALL_CHILD_ERR_EID
Event Message	'Delete All Files error: child task is disabled' 'Delete All Files error: child task queue is full' 'Delete All Files error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-113. (ERROR) EID 46 - 'Delete All Files Error: OS Error'

Internal Name	FM_DELETE_ALL_OS_ERR_EID
Event Message	'Delete All Files error: OS_opendir failed: dir = [dir]' *Where:* • [dir] is the name of the directory or NULL 'Delete All Files error: OS_remove failed: result = [result], file = [file]' *Where:* • [result] is the return value from the OS function call • [file] is the name of the directory entry
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists. Refer to the OS-specific return value for an indication of what might have caused this error.

Table 4-114. (ERROR) EID 48 - 'Decompress File Error: Invalid Command Packet Length'

Internal Name	FM_DECOM_PKT_ERR_EID
Event Message	'Decompress File error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected packet length. • [actual] is the actual packet length.
Cause	This event message is generated upon receipt of a Command 8 - Decompress File (see Table 4-65) telemetry command packet with an invalid length.

Table 4-115. (ERROR) EID 49 - 'Decompress File Error: Source Filename'

Internal Name	FM_DECOM_SRC_ERR_EID
Event Message	'Decompress File error: filename is invalid: name = [name]' 'Decompress File error: file does not exist: name = [name]' 'Decompress File error: file is already open: name = [name]' 'Decompress File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid source filename, • a nonexistent source file, • an open file, or • a pre-existing directory with the same name as the specified source file
Cause	This event message is generated upon receipt of a Command 8 - Decompress File (see Table 4-65) telemetry command packet with an invalid source filename.

Table 4-116. (ERROR) EID 50 - 'Decompress File Error: Target Filename'

Internal Name	FM_DECOM_TGT_ERR_EID
Event Message	'Decompress File error: filename is invalid: name = [name]' 'Decompress File error: file already exists: name = [name]' 'Decompress File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid target filename, • a pre-existing target file, or • a pre-existing directory with the same name as the specified target file
Cause	This event message is generated upon receipt of a Command 8 - Decompress File (see Table 4-65) telemetry command packet with an invalid source filename.

Table 4-117. (ERROR) EID 51 - 'Decompress File Error: Child Task'

Internal Name	FM_DECOM_CHILD_ERR_EID
Event Message	'Decompress File error: child task is disabled' 'Decompress File error: child task queue is full' 'Decompress File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-118. (ERROR) EID 52 - 'Decompress File Error: CFE Error'

Internal Name	FM_DECOM_CFE_ERR_EID
Event Message	'Decompress File error: CFE_FS_Decompress failed: result = [result], src = [src], tgt = [tgt]' *Where:* • [result] is the return value from the API function call • [src] is the source filename • [tgt] is the target filename
Cause	This event message is generated due to an API function error (see Section 2.3.1) that occurred after preliminary command argument verification tests indicated that the source file exists. Refer to the function specific return value for an indication of what might have caused this particular error.

Table 4-119. (ERROR) EID 54 - 'Concat Files Error: Invalid Command Packet Length'

Internal Name	FM_CONCAT_PKT_ERR_EID
Event Message	'Concat Files error: invalid command packet length: expected = [expected] actual =[actual]' *Where:* • [expected] is the expected value of the command packet length • [actual] is the actual value of the command packet length
Cause	This event message is generated upon receipt of a Command 9 - Concatenate Files (see Table 4-66) telemetry packet with an invalid length.

Table 4-120. (ERROR) EID 55 - 'Concat Files Error: Source 1 Filename'

Internal Name	FM_CONCAT_SRC1_ERR_EID
Event Message	'Concat Files error: filename is invalid: name = [name]' 'Concat Files error: file does not exist: name = [name]' 'Concat Files error: file is already open: name = [name]' 'Concat Files error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid source 1 filename, • a nonexistent source 1 file, • an open file, or • a pre-existing directory with the same name as the specified source 1 file
Cause	This event message is generated upon receipt of a Command 9 - Concatenate Files (see Table 4-66) telemetry packet with an invalid source 1 filename.

Table 4-121. (ERROR) EID 56 - 'Concat Files Error: Source 2 Filename'

Internal Name	FM_CONCAT_SRC2_ERR_EID
Event Message	'Concat Files error: filename is invalid: name = [name]' 'Concat Files error: file does not exist: name = [name]' 'Concat Files error: file is already open: name = [name]' 'Concat Files error: filename is a directory: name = [name] ' *Where [name] is, respectively:* • the invalid source 2 filename, • a nonexistent source 2 file, • an open file, or • a pre-existing directory with the same name as the specified source 2 file
Cause	This event message is generated upon receipt of a Command 9 - Concatenate Files (see Table 4-66) telemetry packet with an invalid source 2 filename.

Table 4-122. (ERROR) EID 57 - 'Concat Files Error: Target Filename'

Internal Name	FM_CONCAT_TGT_ERR_EID
Event Message	'Concat Files error: filename is invalid: name = [name]' 'Concat Files error: file already exists: name = [name]' 'Concat Files error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid target filename, • a pre-existing target file, or • a pre-existing directory with the same name as the specified target file
Cause	This event message is generated upon receipt of a Command 9 - Concatenate Files (see Table 4-66) telemetry packet with an invalid target filename.

Table 4-123. (ERROR) EID 58 - 'Concat File Error: Child Task'

Internal Name	FM_CONCAT_CHILD_ERR_EID
Event Message	'Concat File error: child task is disabled' 'Concat File error: child task queue is full' 'Concat File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or some sort of data corruption has affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-124. (ERROR) EID 59 - 'Concat File Error: OS Error'	
Internal Name	FM_CONCAT_OS_ERR_EID
Event Message	'Concat File error: OS_cp failed: result = [result], src = [src], tgt = [tgt]' 'Concat File error: OS_open failed: result = [src2], src2 = [src2]' 'Concat File error: OS_open failed: result = [result], tgt = [tgt]' 'Concat File error: OS_read failed: result = [result], file = [file]' 'Concat File error: OS_write failed: result = [result], expected = [expected]' *Where:* • [result] indicates the actual numeric result code • [expected] indicates the expected numerical result code • [src] indicates the name of the first source file • [src2] indicates the name of the second source file • [tgt] indicates the name of the target file • [file] indicates the filename
Cause	This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Table 4-125. (ERROR) EID 61 - 'Get File Info Error: Invalid Command Packet Length'	
Internal Name	FM_GET_FILE_INFO_PKT_ERR_EID
Event Message	'Get File Info error: invalid command packet length:expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 10 - Get File Information (see Table 4-67) telemetry packet with an invalid length.

Table 4-126. (ERROR) EID 62 - 'Get File Info Error: Source Filename'

Internal Name	FM_GET_FILE_INFO_SRC_ERR_EID
Event Message	'Get File Info error: invalid name: name = [name]' *Where:* • [name] is the invalid filename
Cause	This event message is generated upon receipt of a Command 10 - Get File Information (see Table 4-67) telemetry packet with an invalid filename.

Table 4-127. (ERROR) EID 63 - 'Get File Info Error: Child Task'

Internal Name	FM_GET_FILE_INFO_CHILD_ERR_EID
Event Message	'Get File Info error: child task is disabled' 'Get File Info error: child task queue is full' 'Get File Info error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or some sort of data corruption has affected the interface control variables. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-128. (ERROR) EID 66 - 'Get Open Files Error: Invalid Command Packet Length'

Internal Name	FM_GET_OPEN_FILES_PKT_ERR_EID
Event Message	'Get Open Files error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 11 - Get Open Files Listing (see Table 4-68) telemetry packet with an invalid length.

Table 4-129. (ERROR) EID 68 - 'Create Directory Error: Invalid Command Packet Length'

Internal Name	FM_CREATE_DIR_PKT_ERR_EID
Event Message	'Create Directory error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] indicates the expected command packet length • [actual] indicates the actual command packet length
Cause	This event message is generated upon receipt of a Command 12 - Create Directory (see Table 4-69) telemetry packet with an invalid length.

Table 4-130. (ERROR) EID 69 - 'Create Directory Error: Directory Name'	
Internal Name	FM_CREATE_DIR_SRC_ERR_EID
Event Message	'Create Directory error: directory name is invalid: name = [name]' 'Create Directory error: directory name exists as a file: name = [name]' 'Create Directory error: directory already exists: name = [name]' *Where [name] is, respectively:* • the invalid directory name, • a pre-existing file with the same name as the specified new directory, or • a pre-existing directory with the same name as specified for the new directory
Cause	This event message is generated upon receipt of a Command 12 - Create Directory (see Table 4-69) telemetry packet with an invalid directory name.

Table 4-131. (ERROR) EID 70 - 'Create Directory File Error: Child Task'	
Internal Name	FM_CREATE_DIR_CHILD_ERR_EID
Event Message	'Create Directory File error: child task is disabled' 'Create Directory File error: child task queue is full' 'Create Directory File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the count • [index] is the index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-132. (ERROR) EID 71 - 'Create Directory Error: OS Error'

Internal Name	FM_CREATE_DIR_OS_ERR_EID
Event Message	'Create Directory error: OS_mkdir failed: result = [result], dir = [dir]' *Where:* • [result] is the numeric return value from the OS function call • [dir] is the directory name
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory name is unused and appears to be valid. Refer to the OS specific return value.

Table 4-133. (ERROR) EID 73 - 'Delete Directory Error: Invalid Command Pkt Length'

Internal Name	FM_DELETE_DIR_PKT_ERR_EID
Event Message	'Delete Directory error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 13 - Remove Directory (see Table 4-70) telemetry packet with an invalid length.

Table 4-134. (ERROR) EID 74 - 'Delete Directory Error: Directory Name'

Internal Name	FM_DELETE_DIR_SRC_ERR_EID
Event Message	'Delete Directory error: directory name is invalid: name = [name]' 'Delete Directory error: directory does not exist: name = [name]' 'Delete Directory error: directory name exists as a file: name = [name]' *Where [name] is, respectively:* • the invalid directory name, • a nonexistent directory, or • a pre-existing file with the same name as the specified directory
Cause	This event message is generated upon receipt of a Command 13 - Remove Directory (see Table 4-70) telemetry packet with an invalid directory name.

Table 4-135. (ERROR) EID 75 - 'Delete Directory File Error: Child Task'

Internal Name	FM_DELETE_DIR_CHILD_ERR_EID
Event Message	'Delete Directory File error: child task is disabled' 'Delete Directory File error: child task queue is full' 'Delete Directory File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-136. (ERROR) EID 76 - 'Delete Directory Error: Source Filename'

Internal Name	FM_DELETE_DIR_EMPTY_ERR_EID
Event Message	'Delete Directory error: directory is not empty: dir = [dir]' *Where:* • [dir] is the directory
Cause	This event message is generated upon receipt of a Command 13 - Remove Directory (see Table 4-70) telemetry packet that references a directory that is not empty.

Table 4-137. (ERROR) EID 77 - 'Delete Directory Error: OS Error'

Internal Name	FM_DELETE_DIR_OS_ERR_EID
Event Message	'Delete Directory error: OS_opendir failed: dir = [dir]' 'Delete Directory error: OS_rmdir failed: result = [result], dir = [dir]' *Where:* • [result] is the return value from the OS function call. • [dir] is the name of the directory or the directory entry
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists and appears to be valid. Refer to the OS specific return values. Note: the call to OS_opendir returns a pointer, or NULL.

Table 4-138. (ERROR) EID 79 - 'Directory List to File Error: Invalid Command Packet Length'

Internal Name	FM_GET_DIR_FILE_PKT_ERR_EID
Event Message	'Directory List to File error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 14 - Get Directory Listing to a File (see Table 4-71) telemetry packet with an invalid length.

Table 4-139. (ERROR) EID 80 - 'Directory List to File Error: Directory Name'

Internal Name	FM_GET_DIR_FILE_SRC_ERR_EID
Event Message	'Directory List to File error: directory name is invalid: name = [name]' 'Directory List to File error: directory does not exist: name = [name]' 'Directory List to File error: directory name exists as a file: name = [name]' *Where [name] is, respectively:* • the invalid directory name, • a nonexistent directory, or • a pre-existing file with the same name as the specified directory
Cause	This event message is generated upon receipt of a Command 14 - Get Directory Listing to a File (see Table 4-71) telemetry packet with an invalid directory name.

Table 4-140. (ERROR) EID 81 - 'Directory List to File Error: Output Filename'

Internal Name	FM_GET_DIR_FILE_TGT_ERR_EID
Event Message	'Directory List to File error: filename is invalid: name = [name]' 'Directory List to File error: file already exists: name = [name]' 'Directory List to File error: filename is a directory: name = [name]' *Where [name] is, respectively:* • the invalid output filename, • a pre-existing file with the same name, or • a pre-existing directory with the same name as the specified output file
Cause	This event message is generated upon receipt of a Command 14 - Get Directory Listing to a File (see Table 4-71) telemetry packet with an invalid output filename.

Table 4-141. (ERROR) EID 83 - 'Directory List to File Error: Child Task'	
Internal Name	FM_GET_DIR_FILE_CHILD_ERR_EID
Event Message	'Directory List to File error: child task is disabled' 'Directory List to File error: child task queue is full' 'Directory List to File error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-142. (ERROR) EID 84 - 'Directory List to File Error: OS Error'

Internal Name	FM_GET_DIR_FILE_OS_ERR_EID
Event Message	'Directory List to File error: OS_opendir failed: dir = [dir]' 'Directory List to File error: OS_write blank stats failed: result = [result], expected =[expected]’ 'Directory List to File error: CFE_FS_WriteHeader failed: result = [result], expected =[expected]' 'Directory List to File error: OS_creat failed: result = [result], file = [file]' 'Directory List to File error: OS_write entry failed: result = result = [result], expected =[expected]' 'Directory List to File error: OS_write update stats failed: result = [result], expected =[expected]’ *Where:* • [result] and [expected] are the return values from the OS function call. • [dir] and [file] are the names of the directory and directory entry, respectively.
Cause	This event message is generated due to an OS function error that occurred after preliminary command argument verification tests indicated that the directory exists and the output filename is unused and appears to be valid. Verify that the output filename is reasonable. Also, verify that the file system has sufficient free space for this operation. Then refer to the OS specific return values. Note: the call to OS_opendir returns a pointer, or NULL.

Table 4-143. (ERROR) EID 87 - 'Directory List to Packet Error: Invalid Command Packet Length'

Internal Name	FM_GET_DIR_PKT_PKT_ERR_EID
Event Message	'Directory List to Packet error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 15 - Get Directory Listing to a Packet (see Table 4-72) telemetry packet with an invalid length.

Table 4-144. (ERROR) EID 88 - 'Directory List to Packet Error: Directory Name'

Internal Name	FM_GET_DIR_PKT_SRC_ERR_EID
Event Message	'Directory List to Packet error: directory name is invalid: name = [name]' 'Directory List to Packet error: directory does not exist: name = [name]' 'Directory List to Packet error: directory name exists as a file: name = [name]' *Where [name] is, respectively:* • the invalid directory name, • a nonexistent directory, or • a pre-existing file with the same name as the specified directory
Cause	This event message is generated upon receipt of a Command 15 - Get Directory Listing to a Packet (see Table 4-72) telemetry packet with an invalid directory name.

Table 4-145. (ERROR) EID 89 - 'Directory List to Packet Error: Child Task'

Internal Name	FM_GET_DIR_PKT_CHILD_ERR_EID
Event Message	'Directory List to Packet error: child task is disabled' 'Directory List to Packet error: child task queue is full' 'Directory List to Packet error: child task interface is broken: count = [count], index = [index]' *Where:* • [count] is the numeric count • [index] is the numeric index
Cause	This event message is generated when the FM child task command queue interface cannot be used. If the child task command queue is full, the problem may be temporary, caused by sending too many FM commands too quickly. If the command queue does not empty itself within a reasonable amount of time then the child task may be hung. It may be possible to use cFE commands to terminate the child task, which should then cause FM to process all commands in the main task. If the child task queue is broken, either the handshake interface logic is flawed, or the interface control variables have been affected by some sort of data corruption. In either case, it may be necessary to restart the FM application to resync the interface.

Table 4-146. (ERROR) EID 90 - 'Directory List to Packet Error: OS Error'

Internal Name	FM_GET_DIR_PKT_OS_ERR_EID
Event Message	'Directory List to Packet error: OS_opendir failed: dir = [dir]' 'Directory List to Packet error: ' *Where:* • [dir] is the name of the directory or the directory entry.
Cause	The call to OS_opendir returns either a pointer or NULL.

Table 4-147. (ERROR) EID 92 - 'Get Free Space Error: Invalid Command Packet Length'

Internal Name	FM_GET_FREE_SPACE_PKT_ERR_EID
Event Message	'Get Free Space error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the command packet length • [actual] is the actual packet length
Cause	This event message is generated upon receipt of a Command 16 - Get Free Space (see Table 4-73) telemetry packet with an invalid length.

Table 4-148. (ERROR) EID 93 - 'Get Free Space Error: Table Not Loaded'

Internal Name	FM_GET_FREE_SPACE_TBL_ERR_EID
Event Message	'Get Free Space error: file system free space table is not loaded'
Cause	This event message is generated upon receipt of a Command 16 - Get Free Space (see Table 4-73) telemetry packet when the FM file system free space table has not yet been loaded.

Table 4-149. (ERROR) EID 95 - 'Set Table State Error: Invalid Command Packet Length'

Internal Name	FM_SET_TABLE_STATE_PKT_ERR_EID
Event Message	'Set Table State error: invalid command packet length: expected = [expected], actual = [actual]' *Where:* • [expected] is the command packet length • [actual] is the actual packet length
Cause	This event message is generated upon receipt of a Command 17 - Set Free Space Table Entry State (see Table 4-74) telemetry packet with an invalid length.

Table 4-150. (ERROR) EID 96 - 'Set Table State Error: Table Not Loaded'

Internal Name	FM_SET_TABLE_STATE_TBL_ERR_EID
Event Message	'Set Table State error: file system free space table is not loaded'
Cause	This event message is generated upon receipt of a Command 17 - Set Free Space Table Entry State (see Table 4-74) telemetry packet when the FM file system free space table has not yet been loaded.

Table 4-151. (ERROR) EID 97 - 'Set Table State Error: Invalid Command Argument'

Internal Name	FM_SET_TABLE_STATE_ARG_ERR_EID
Event Message	'Set Table State error: invalid command argument: index = [index]' 'Set Table State error: invalid command argument: state = [state]' *Where:* • [index] points to the invalid table index • [state] is the invalid entry state argument
Cause	This event message is generated upon receipt of a Command 17 - Set Free Space Table Entry State (see Table 4-74) telemetry packet with an invalid table index or entry state argument.

Table 4-152. (ERROR) EID 98 - 'Set Table State Error: Unused Table Entry'

Internal Name	FM_SET_TABLE_STATE_UNUSED_ERR_EID
Event Message	'Set Table State error: cannot modify unused table entry: index = [index]' *Where:* • [index] points to the unused free space table entry
Cause	This event message is generated upon receipt of a Command 17 - Set Free Space Table Entry State (see Table 4-74) telemetry packet that references an unused free space table entry.

Table 4-153. (ERROR) EID 99 - 'Free Space Table Verify Error'

Internal Name	FM_TABLE_VERIFY_ERR_EID
Event Message	'Free Space Table verify error: index = [index], empty name string' 'Free Space Table verify error: index = [index], name too long' 'Free Space Table verify error: index = [index], invalid name = [name]' 'Free Space Table verify error: index = [index], invalid state = [state]' *Where:* • [index] is the pointer to the file system free space table record • [name] is the file system name • [state] is the table entry state (enabled or disabled)
Cause	This event message is generated when a file system free space table fails the table verification process. Each file system table entry has only two fields: table entry state and file system name. The table entry state field must be either enabled or disabled. The file system name string must have a non-zero length, include a string terminator, and not contain characters considered invalid for filenames. If the file system free space table loaded at startup fails verification, the FM application will not terminate. However, if a file system free space table has not been successfully loaded, the FM application will not process commands that request the file system free space telemetry packet. Thereafter, if an attempt to load a new table fails verification, the FM application will continue to use the previous table.

Table 4-154. (ERROR) EID 101 - 'Child Task Initialization Error'

Internal Name	FM_CHILD_INIT_ERR_EID
Event Message	'Child Task initialization error: create task failed: result =[result]' ‘Child Task initialization error: register child failed: result =[result]' ‘Child Task initialization error: create semaphore failed: result =[result]' *Where:* • [result] is the return value from the OS function call
Cause	This event message indicates an unsuccessful attempt to create and initialize the low priority FM child task. Commands which normally would hand off to the child task for execution will now be processed by the main FM application. Refer to the return code in the event text for the exact cause of the error.

Table 4-155. (ERROR) EID 102 - 'Child Task Termination Error'

Internal Name	FM_CHILD_TERM_ERR_EID
Event Message	'Child Task termination error: empty queue' 'Child Task termination error: invalid queue index: index = [index]' 'Child Task termination error: semaphore take failed: result = [result]' *Where:* • [index] is pointer to the task queue • [result] is the event return value from the function call
Cause	This event message indicates that the FM child task has suffered a fatal error and has terminated. The error occurred when referencing the interface handshake variables or while pending on the handshake semaphore. Refer to the event text for the exact cause of the error.

Table 4-156. (ERROR) EID 103 - 'Child Task Execution Error'	
Internal Name	FM_CHILD_EXE_ERR_EID
Event Message	'Child Task execution error: invalid command code: cc = [CC]' *Where:* • [CC] indicates the invalid command code passed from the handshake request call.
Cause	This event message indicates that the FM child task is unable to process the current handshake request. Either the handshake queue index or the handshake command code is invalid. This error suggests that either the handshake interface logic is flawed, or there has been some sort of data corruption that affected the interface data. It may be necessary to restart the FM application to resync the handshake interface.

Table 4-157. (ERROR) EID 110 - 'Set Permissions error: Invalid Command Packet Length'	
Internal Name	FM_SET_PERM_ERR_EID
Event Message	'Set Permissions error: invalid command packet length:expected = [expected], actual = [actual]' *Where:* • [expected] is the expected command packet length • [actual] is the actual command packet length
Cause	This event message is generated upon receipt of a Command 19 - Set File Permissions (see Table 4-76) telemetry packet with an invalid length.

Table 4-158. (ERROR) EID 112 - 'Set Permissions: OS_chmod Error'	
Internal Name	FM_SET_PERM_OS_ERR_EID
Event Message	'Set Permissions: OS_chmod error, RC=0x08X, file = [file], access = [access]' *Where:* • [file] is the name of the file • [access] is the modified access requested by the command, the "Mode" parameter in the FM_SetPermCmd_t command structure
Cause	This event message is generated upon receipt of a Command 19 - Set File Permissions (see Table 4-76) telemetry packet with an OS_chmod Error.

4.4.3 Event Messages - INFORMATION

For an index of all event messages, see Appendix A .

cFS FM contains nine **info** event messages:

1. EID 1 - 'Initialization Complete' (see Table 4-159)
2. EID 12 - 'No-op Command Success' (see Table 4-160)
3. EID 42 - 'Delete All Files Warning' (see Table 4-161)
4. EID 64 - 'Get File Info Warning: unable to compute CRC' (see Table 4-162)
5. EID 82 - 'Directory List to File Warning: Pathname' (see Table 4-163)
6. EID 86 - 'Directory List to Packet Warning' (see Table 4-164)
7. EID 94 - 'Set Table State Command' (see Table 4-165)
8. EID 100 - 'Child Task Initialization Complete' (see Table 4-166)
9. EID 104 - 'Free Space Table Validation Results' (see Table 4-167)

Table 4-159. (INFO) EID 1 - 'Initialization Complete'

Internal Name	FM_STARTUP_EID
Event Message	'Initialization complete: version [Major].[Minor].[Revision].[Mission]' *Where:* • [Major] is the major version identifier • [Minor] is the minor version identifier • [Revision] is the revision identifier • [Mission] is the mission revision identifier.
Cause	This event message is issued after the File Manager application has successfully completed startup initialization.

Table 4-160. (INFO) EID 12 - 'Noop Command Success'

Internal Name	FM_NOOP_CMD_EID
Event Message	'No-op command: Version [Major].[Minor].[Revision].[Mission]' *Where:* • [Major] is the major version identifier • [Minor] is the minor version identifier • [Revision] is the revision identifier • [Mission] is the mission revision identifier.
Cause	This event message signals the successful completion of Command 0 - No Operation (see Table 4-58). The version data includes the application major version, minor version, revision, and mission revision numbers.

Table 4-161. (INFO) EID 42 - 'Delete All Files Warning'

Internal Name	FM_DELETE_ALL_WARNING_EID
Event Message	'Delete All Files warning: combined directory and entry name too long: dir = [dir], entry = [entry]' 'Delete All Files warning: entry is invalid: entry = [entry]' 'Delete All Files warning: entry no longer exists: entry = [entry]' 'Delete All Files warning: cannot delete subdirectory: sub = [sub]' 'Delete All Files warning: cannot delete open file: file = [file]' 'Delete All Files warning: OS_remove failed: result = [result], file = [file]' *Where:* • [dir] is the name of the directory • [entry] is the name of the entry • [sub] is the name of the subdirectory • [file] is the name of the file • [result] is the numeric returned from the OS
Cause	Command 7 - Delete All Files (see Table 4-64) will succeed if the handler is successfully able to read the directory and attempt to delete the entries in the directory. Command warnings are issued when directory entries, for whatever reason, cannot be deleted.

Table 4-162. (INFO) EID 64 - 'Get File Info Warning: Unable to Compute CRC'

Internal Name	FM_GET_FILE_INFO_WARNING_EID
Event Message	'Get File Info warning: unable to compute CRC: invalid file state = [numeric], file = [file]' 'Get File Info warning: unable to compute CRC: invalid CRC type = [numeric], file = [file]' 'Get File Info warning: unable to compute CRC: OS_open result = [numeric], file = [file]' 'Get File Info warning: unable to compute CRC: OS_read result = [numeric], file = [file]' *Where:* • [numeric] is the return value from the API function call. • [file] is the name of the file being operated on by the API function.
Cause	This event message is generated due to an API function error that occurred after preliminary command argument verification tests indicated that the source files exist. Refer to the function specific return value for an indication of what might have caused this particular error.

Table 4-163. (INFO) EID 82 - 'Directory List to File Warning: Pathname'

Internal Name	FM_GET_DIR_FILE_WARNING_EID
Event Message	'Directory List to File warning: combined directory and entry name too long: dir =[dir], entry = [entry]' *Where:* • [dir] is the directory name • [entry] is the entry name
Cause	This event message is generated when the combined length of the directory name plus the directory entry name exceeds the maximum qualified filename length. It is unclear how this condition might arise, but since we are copying both strings into a fixed length buffer, we must first verify the length. The Command 14 - Get Directory Listing to a File (see Table 4-71) command handler will not write information regarding this directory entry to the output file.

Table 4-164. (INFO) EID 86 - 'Directory List to Packet Warning'

Internal Name	FM_GET_DIR_PKT_WARNING_EID
Event Message	'Directory List to Packet warning: dir + entry is too long: dir = [dir], entry = [entry]' *Where:* • [dir] is the directory name • [entry] is the directory entry name
Cause	This event message is generated when the combined length of the directory name plus the directory entry name exceeds the maximum qualified filename length. It is unclear how this condition might arise, but when copying both strings into a fixed length buffer, the length must first be verified.

Table 4-165. (INFO) EID 94 - 'Set Table State Command'

Internal Name	FM_SET_TABLE_STATE_CMD_EID
Event Message	'Set Table State command: index = [index], state = [state]' *Where:* • [index] points to the table entry • [state] is the 32 bit unsigned integer representing the state of the file
Cause	This event message signals the successful completion of Command 17 - Set Free Space Table Entry State (see Table 4-74).

Table 4-166. (INFO) EID 100 - 'Child Task Initialization Complete'

Internal Name	FM_CHILD_INIT_EID
Event Message	'Child Task initialization complete'
Cause	This event message signals the successful completion of the initialization process for the FM child task.

Table 4-167. (INFO) EID 104 - 'Free Space Table Validation Results'	
Internal Name	FM_TABLE_VERIFY_EID
Event Message	'Free Space Table verify results: good = [good], bad = [bad], unused = [unused]' *Where:* • [good] is the number of table entries that were verified without error • [bad] is the number of table entries that had one or more errors • [unused] is the number of unused table entries
Cause:	This event describes the results of the Free Space Table validation function. The cFE Table Services Manager will call this function autonomously when the default table is loaded at startup. The Manager will also call this function autonomously when a table validate command that targets this table is processed. The sum of good + bad + unused results will equal the total number of table entries.

4.4.4 Event Messages - DEBUG

For an index of all event messages, see Appendix A .

The tables in this section show **debug** event messages for cFS FM. Events of this type are primarily for the mission developer. The messages contain specific references to code and are of limited use to flight operators. By default these debug messages are disabled. The operator/tester/user/developer must command them enabled.

cFS FM contains 16 debug event messages:

1. EID 14 - 'Reset Counters Command Success' (see Table 4-168)
2. EID 16 - 'Copy File Command Success' (see Table 4-169)
3. EID 23 - 'Move File Command Success' (see Table 4-170)
4. EID 30 - 'Rename File Command Success' (see Table 4-171)
5. EID 36 - 'Delete File Command Success' (see Table 4-172)
6. EID 41 - 'Delete All Files Command Success' (see Table 4-173)
7. EID 47 - 'Decompress File Command Success' (see Table 4-174)
8. EID 53 - 'Concat Files Command Success' (see Table 4-175)
9. EID 60 - 'Get File Info Command Success' (see Table 4-176)
10. EID 65 - 'Get Open Files Command Success' (see Table 4-177)
11. EID 67 - 'Create Directory Command Success' (see Table 4-178)
12. EID 72 - 'Delete Directory Command Success' (see Table 4-179)
13. EID 78 - 'Directory List to File command' (see Table 4-180)
14. EID 85 - 'Directory List to Packet command' (see Table 4-181)
15. EID 91 - 'Get Free Space Command' (see Table 4-182)
16. EID 111 - 'Set Permissions Command' (see Table 4-183)

Table 4-168. (DEBUG) EID 14 - 'Reset Counters Command Success'	
Internal Name	FM_RESET_CMD_EID
Event Message	'Reset Counters command'
Cause	This event message signals the successful completion of Command 1 - Reset Counters (see Table 4-59).

Table 4-169. (DEBUG) EID 16 - 'Copy File Command Success'	
Internal Name	FM_COPY_CMD_EID
Event Message	'Copy File command: src = [src], tgt = [tgt]' *Where:* • [src] is the name of the source file • [tgt] is the name of the target file
Cause	This event message signals the successful completion of Command 2 - Copy File (see Table 4-60). Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.

Table 4-170. (DEBUG) EID 23 - 'Move File Command Success'	
Internal Name	FM_MOVE_CMD_EID
Event Message	'Move File command: src = [src], tgt = [tgt]' *Where:* • [src] is the name of the source file • [tgt] is the name of the target file
Cause	This event message signals the successful completion of Command 3 - Move File (see Table 4-61).

Table 4-171. (DEBUG) EID 30 - 'Rename File Command Success'	
Internal Name	FM_RENAME_CMD_EID
Event Message	'Rename File command: src = [src], tgt = [tgt]' *Where:* • [src] is the name of the source file • [tgt] is the name of the target file
Cause	This event message signals the successful completion of Command 4 - Rename File (see Table 4-62).

Table 4-172. (DEBUG) EID 36 - 'Delete File Command Success'

Internal Name	FM_DELETE_CMD_EID
Event Message	'Delete File command: file = [file]' *Where:* • [file] is the name of the deleted file
Cause	This event message signals the successful completion of Command 5 - Delete File (see Table 4-63).

Table 4-173. (DEBUG) EID 41 – Delete All Files Command Success

Internal Name	FM_DELETE_ALL_CMD_EID
Event Message	'Delete All Files command: deleted [deleted] of [total] dir entries: dir = [dir]' *Where:* • [deleted] indicates the number of files deleted • [total] indicates the total number of files • [dir] indicates the name of the directory
Cause	This event message signals the successful completion of Command 7 - Delete All Files (see Table 4-64). Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.

Table 4-174. (DEBUG) EID 47 – Decompress File Command Success

Internal Name	FM_DECOM_CMD_EID
Event Message	'Decompress File command: src = [src], tgt = [tgt]' *Where:* • [src] is the name of the source file • [tgt] is the name of the target file
Cause	This event message signals the successful completion of Command 8 - Decompress File (see Table 4-65). Note that the execution of this command generally occurs within the context of the cFS FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.

Table 4-175. (DEBUG) EID 53 – Concat Files Command Success

Internal Name	FM_CONCAT_CMD_EID
Event Message	'Concat Files command: src1 = [src1], src2 = [src2], tgt = [tgt]' *Where:* • [src1] is the name of the first source file • [src2] is the name of the second source file • [tgt] is the name of the target file
Cause	This event message signals the successful completion of Command 9 - Concatenate Files (see Table 4-66). Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.

Table 4-176. (DEBUG) EID 60 - 'Get File Info Command Success'	
Internal Name	FM_GET_FILE_INFO_CMD_EID
Event Message	'Get File Info command: name = [name]' *Where:* • [name] is the name of the targeted file
Cause	This event message signals the successful completion of Command 10 - Get File Information (see Table 4-67). Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.
Table 4-177. (DEBUG) EID 65 - 'Get Open Files Command Success'	
Internal Name	FM_GET_OPEN_FILES_CMD_EID
Event Message	'Get Open Files command'
Cause	This event message signals the successful completion of Command 11 - Get Open Files Listing (see Table 4-68).
Table 4-178. (DEBUG) EID 67 - 'Create Directory Command Success'	
Internal Name	FM_CREATE_DIR_CMD_EID
Event Message	'Create Directory command: dir = [dir]' *Where:* • [dir] is the name of the created directory
Cause	This event message signals the successful completion of Command 12 - Create Directory (see Table 4-69).

Table 4-179. (DEBUG) EID 72 - 'Delete Directory Command Success'	
Internal Name	FM_DELETE_DIR_CMD_EID
Event Message	'Delete Directory command: dir = [dir]' *Where:* • [dir] is the name of the deleted directory
Cause	This event message signals the successful completion of Command 13 - Remove Directory (see Table 4-70).

Table 4-180. (DEBUG) EID 78 - 'Directory List to File command'	
Internal Name	FM_GET_DIR_FILE_CMD_EID
Event Message	'Directory List to File command: wrote [written] of [total_names] names: dir = [dir], filename = [filename]' *Where:* • [written] is the number of directory list entries successfully saved to the file • [total_names] is the total number of directory list entries • [dir] is the name of the source directory • [filename] is the name of the target file
Cause	This event message signals the successful completion of Command 14 - Get Directory Listing to a File (see Table 4-71). Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command. Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.

Table 4-181. (DEBUG) EID 85 - 'Directory List to Packet command'

Internal Name	FM_GET_DIR_PKT_CMD_EID
Event Message	'Directory List to Packet command: offset = [offset], dir = [dir]' *Where:* • [offset] is the an index argument defining which entry in the directory is the first entry reported in the telemetry packet • [dir] is the name of the directory
Cause	This event message signals the successful completion of Command 15 - Get Directory Listing to a Packet (see Table 4-72). Note that the execution of this command generally occurs within the context of the FM low priority child task. Thus, this event may not occur until sometime after the command was invoked. However, this event message does signal the actual completion of the command.

Table 4-182. (DEBUG) EID 91 - 'Get Free Space Command'

Internal Name	FM_GET_FREE_SPACE_CMD_EID
Event Message	'Get Free Space command'
Cause	This event message signals the successful completion of Command 16 - Get Free Space (see Table 4-73). This event is type debug because the command generates a telemetry packet that also signals the completion of the command.

Table 4-183. (DEBUG) EID 111 - 'Set Permissions Command'

Internal Name	FM_SET_PERM_CMD_EID
Event Message	'Set Permissions command: filename = [filename], state = [state]' *Where:* • [filename] is the name of the file • [state] is the 32 bit unsigned integer representing the state of the file
Cause	This event message signals the successful completion of Command 19 - Set File Permissions (see Table 4-76).

Appendix A Event Message Cross Reference

This appendix lists all events by EID:

Table A-1. Messages by EID

EID	Type	Message	Detail
EID 001	Info	'Initialization Complete'	Table 4-159
EID 002	Error	'Initialization Error: Register for Event Services'	Table 4-77
EID 003	Error	'Initialization Error: Create SB Input Pipe'	Table 4-78
EID 004	Error	'Initialization Error: Subscribe to HK Request'	Table 4-79
EID 005	Error	'Initialization Error: Subscribe to FM Commands'	Table 4-80
EID 006	Error	'Initialization Error: Register Free Space Table'	Table 4-81
EID 007	Error	'Main Loop Error: Software Bus Receive'	Table 4-82
EID 008	Error	'Application Terminating'	Table 4-83
EID 009	Error	'Main Loop Error: Invalid Message ID'	Table 4-84
EID 010	Error	'Main Loop Error: Invalid Command Code'	Table 4-85
EID 011	Error	'HK Request Error: Invalid Command Packet Length'	Table 4-86
EID 012	Info	'No-op Command Success'	Table 4-160
EID 013	Error	'No-op Error: Invalid Command Packet Length'	Table 4-87
EID 014	Debug	'Reset Counters Command Success'	Table 4-168
EID 015	Error	'Reset Counters Error: Invalid Command Packet Length'	Table 4-88
EID 016	Debug	'Copy File Command Success'	Table 4-169
EID 017	Error	'Copy File Error: Invalid Command Packet Length'	Table 4-89
EID 018	Error	'Copy File Error: Invalid Overwrite'	Table 4-90

EID	Type	Message	Detail
EID 019	Error	'Copy File Error: Source Filename'	Table 4-91
EID 020	Error	'Copy File Error: Target Filename'	Table 4-92
EID 021	Error	'Copy File Error: Child Task'	Table 4-93
EID 022	Error	'Copy File Error: OS Error'	Table 4-94
EID 023	Debug	'Move File Command Success'	Table 4-170
EID 024	Error	'Move File Error: Invalid Command Packet Length'	Table 4-95
EID 025	Error	'Move File Error: Invalid Overwrite'	Table 4-96
EID 026	Error	'Move File Error: Source Filename'	Table 4-97
EID 027	Error	'Move File Error: Target Filename'	Table 4-98
EID 028	Error	'Move File Error: Child Task'	Table 4-99
EID 029	Error	'Move File Error: OS Error'	Table 4-100
EID 030	Debug	'Rename File Command Success'	Table 4-171
EID 031	Error	'Rename File Error: Invalid Command Packet Length'	Table 4-101
EID 032	Error	'Rename File Error: Source Filename'	Table 4-102
EID 033	Error	'Rename File Error: Target Filename'	Table 4-103
EID 034	Error	'Rename File Error: Child Task'	Table 4-104
EID 035	Error	'Rename File Error: OS Error'	Table 4-105
EID 036	Debug	'Delete File Command Success'	Table 4-172
EID 037	Error	'Delete File Error: Invalid Command Packet Length'	Table 4-106
EID 038	Error	'Delete File Error: Filename'	Table 4-107
EID 039	Error	'Delete File Error: Child Task'	Table 4-108
EID 040	Error	'Delete File Error: OS Error'	Table 4-109
EID 041	Debug	'Delete All Files Command Success'	Table 4-173
EID 042	Info	'Delete All Files Warning'	Table 4-161

EID	Type	Message	Detail
EID 043	Error	'Delete All Files Error: Invalid Command Packet Length'	Table 4-110
EID 044	Error	'Delete All Files Error: Directory Name'	Table 4-111
EID 045	Error	'Delete All Files Error: Child Task'	Table 4-112
EID 046	Error	'Delete All Files Error: OS Error'	Table 4-113
EID 047	Debug	'Decompress File Command Success'	Table 4-174
EID 048	Error	'Decompress File Error: Invalid Command Packet Length'	Table 4-114
EID 049	Error	'Decompress File Error: Source Filename'	Table 4-115
EID 050	Error	'Decompress File Error: Target Filename'	Table 4-116
EID 051	Error	'Decompress File Error: Child Task'	Table 4-117
EID 052	Error	'Decompress File Error: CFE Error'	Table 4-118
EID 053	Debug	'Concat Files Command Success'	Table 4-175
EID 054	Error	'Concat Files Error: Invalid Command Packet Length'	Table 4-119
EID 055	Error	'Concat Files Error: Source 1 Filename'	Table 4-120
EID 056	Error	'Concat Files Error: Source 2 Filename'	Table 4-121
EID 057	Error	'Concat Files Error: Target Filename'	Table 4-122
EID 058	Error	'Concat File Error: Child Task'	Table 4-123
EID 059	Error	'Concat File Error: OS Error'	Table 4-124
EID 060	Debug	'Get File Info Command Success'	Table 4-176
EID 061	Error	'Get File Info Error: Invalid Command Packet Length'	Table 4-125
EID 062	Error	'Get File Info Error: Source Filename'	Table 4-126
EID 063	Error	'Get File Info Error: Child Task'	Table 4-127
EID 064	Info	'Get File Info Warning: unable to compute CRC'	Table 4-162
EID 065	Debug	'Get Open Files Command Success'	Table 4-177

EID	Type	Message	Detail
EID 066	Error	'Get Open Files Error: Invalid Command Packet Length'	Table 4-128
EID 067	Debug	'Create Directory Command Success'	Table 4-178
EID 068	Error	'Create Directory Error: Invalid Command Packet Length'	Table 4-129
EID 069	Error	'Create Directory Error: Directory Name'	Table 4-130
EID 070	Error	'Create Directory File Error: Child Task'	Table 4-131
EID 071	Error	'Create Directory Error: OS Error'	Table 4-132
EID 072	Debug	'Delete Directory Command Success'	Table 4-179
EID 073	Error	'Delete Directory Error: Invalid Command Packet Length'	Table 4-133
EID 074	Error	'Delete Directory Error: Directory Name'	Table 4-134
EID 075	Error	'Delete Directory File Error: Child Task'	Table 4-135
EID 076	Error	'Delete Directory Error: Source Filename'	Table 4-136
EID 077	Error	'Delete Directory Error: OS Error'	Table 4-137
EID 078	Debug	'Directory List to File command'	Table 4-180
EID 079	Error	'Directory List to File Error: Invalid Command Packet Length'	Table 4-138
EID 080	Error	'Directory List to File Error: Directory Name'	Table 4-139
EID 081	Error	'Directory List to File Error: Output Filename'	Table 4-140
EID 082	Info	'Directory List to File Warning: Pathname'	Table 4-163
EID 083	Error	'Directory List to File Error: Child Task'	Table 4-141
EID 084	Error	'Directory List to File Error: OS Error'	Table 4-142
EID 085	Debug	'Directory List to Packet command'	Table 4-181
EID 086	Info	'Directory List to Packet Warning'	Table 4-164
EID 087	Error	'Directory List to Packet Error: Invalid Command Packet Length'	Table 4-143

EID	Type	Message	Detail
EID 088	Error	'Directory List to Packet Error: Directory Name'	Table 4-144
EID 089	Error	'Directory List to Packet Error: Child Task'	Table 4-145
EID 090	Error	'Directory List to Packet Error: OS Error'	Table 4-146
EID 091	Debug	'Get Free Space Command'	Table 4-182
EID 092	Error	'Get Free Space Error: Invalid Command Packet Length'	Table 4-147
EID 093	Error	'Get Free Space Error: Table Not Loaded'	Table 4-148
EID 094	Info	'Set Table State Command'	Table 4-165
EID 095	Error	'Set Table State Error: Invalid Command Packet Length'	Table 4-149
EID 096	Error	'Set Table State Error: Table Not Loaded'	Table 4-150
EID 097	Error	'Set Table State Error: Invalid Command Argument'	Table 4-151
EID 098	Error	'Set Table State Error: Unused Table Entry'	Table 4-152
EID 099	Error	'Free Space Table Verify Error'	Table 4-153
EID 100	Info	'Child Task Initialization Complete'	Table 4-166
EID 101	Error	'Child Task Initialization Error'	Table 4-154
EID 102	Error	'Child Task Termination Error'	Table 4-155
EID 103	Error	'Child Task Execution Error'	Table 4-156
EID 104	Info	'Free Space Table Validation Results'	Table 4-167
EID 110	Error	'Set Permissions error: Invalid Command Packet Length'	Table 4-157
EID 111	Debug	'Set Permissions Command'	Table 4-183
EID 112	Error	'Set Permissions: OS_chmod error'	Table 4-158

Appendix B **Document Notes**

This appendix is for those who need to edit this document.

NOTE: Before editing this document, change MS word options to display hidden text, to avoid accidentally deleting TOC heading entries.

B.1 Mission-Specific Conventions

- This document presents in an *orange font* selected information that should be replaced when tailoring this document for a mission.
- Specifically, this document as delivered to the mission has names that may be replaced by the mission when the mission creates the ground system RDL database. In particular, the suggested names start with $sc_$CPU_FM which indicate a global setting for spacecraft, processor selection, and the cFS FM subsystem. This has meaning if the mission has multiple spacecraft, each with a copy of cFE/cFS applications being executed, and/or multiple processors per spacecraft, each with a copy of cFE/cFS applications being executed. Most missions have neither and they do not prepend a $sc_$cpu_ selection to the front of the command name. However, it is common for missions to differentiate the spacecraft subsystem commands from instrument commands by prepending several characters (e.g. pw for power system) to all the command and telemetry names for that subsystem.

B.2 Tailoring this Document for Missions

This section is for anyone updating this Guide, either using it as a template for other applications, or tailoring it for mission specific use.

NOTE: Use caution when updating table captions in Section 4.3. For consistency, all command names throughout the document use pointers to these captions. Changing the caption will change the name of command references throughout the document.

When tailoring this document for a particular mission, replace text appearing *in this orange font.*

- Review figures to be sure that there is no conflict with mission configurations. Edit figures with Microsoft Visio or PowerPoint if necessary.
- Add mission-defined values in Chapter 4.
- Regenerate the table of contents. TOC, Figures, and Tables can be updated automatically. To update all figure and table references in the document, when using the PC version of Word, select all, then choose F9.
- Update document tags and other document metadata (File, Info).

This Guide is formatted using Microsoft Word styles. When adding new sections to the Guide, tag material with the styles shown below. Center all figures horizontally on the page. To automatically format tables, use the Word Design, Table Styles menu and assign the custom styles as shown under Table Styles in Table B-1 below.

Table of Contents subheadings that do not have page numbers are generally inserted as Table of Contents (TC) codes in the text in which the content appears, usually at the end of a paragraph before the <RETURN> character. In the Table of Contents at the beginning of the document, the \f switch has been added to the Table of Tables hidden codes to make the subheadings visible.

Margins are set to Normal, i.e., one inch top, one inch bottom, one inch left, and one inch right.

Table B-1. Styles in this Document

Type	Style to be Used
Chapter titles, subtitles, and subsections.	“Heading 1” through “Heading 6”
Appendix title	Heading 7 (“Appendix A Head”)
Appendix headings 1	Heading 8 (“A.1 Heading 8”)
Appendix headings 2	Heading 9 (“A.1.1 Heading 9”)
Names of code modules, file names, code snippets	“Code”
All text not otherwise tagged	“Body Text”
Cover styles	“Cover_blank” “Cover_cfs” “Cover_doc_title” “Cover_app_user_guide” “Cover_footer” “Cover_fsb” “Cover_meatball” “Cover_mIssion_name” “Cover_normal_right”
Tables & Figures	
Bullets in tables, flush left	“Table Bullets Indent 0”
Bullets in tables, single indent	“Table Bullets Indent 1”
Bullets in tables, double indent	“Table Bullets Indent 2”
Heading above double indented table bullets	“Table Bullets Head”
Figure captions	“Caption Figure”
Table captions	“Caption Table”
cFS default values	“cFS Default”
Table Styles	
Tables with horizontal headers	“cFS Tables”
Tables with vertical left headers	“cFS Sideways Tables”
In hybrid tables (multiple captioned tables that are combined into one long table), all rows are tagged “Widow/Orphan control” (paragraph menu). Additionally, to keep each captioned section on the same page, all but the last row in each captioned section is tagged “Keep with next” (paragraph menu).	