Version 1.2 November 3, 2010

CFS CF Requirements

Code 582
Flight Software Systems Branch

Nov 3, 2010

ID	Summary
9878	CFS CF Requirements Document

SR_Contains:

ID	ReqID	Text	Rationale	Heritage_Reference
9880		**CFS CCSDS File Delivery Protocol (CF) Requirements**		
9882		**1.0 Introduction** **1.1 Document Purpose** The Core Flight System (CFS) CCSDS File Delivery Protocol (CFDP) Application (also referenced as CF Application) will be developed by the Flight Software Branch (FSB) of the Software Engineering Division (SED). The purpose of this requirements specification is to define the requirements to be satisfied by the CF Application. This application is developed for re-use. For this reason, several nomenclatures are used in this document to identify configurations for a mission. The CFS is specified as a multi-platform product. Mission-specific features and customization requirements, which are applicable for all platforms, are tagged with <MISSION_DEFINED>. Platform-specific features and customizations requirements are tagged with either "<PLATFORM_DEFINED>" or "<OPTIONAL>." Additional nomenclature is used along with the tag to specify a CFS default value for the platform-specific feature: "<PLATFORM_DEFINED, Default_Value>". Reference platforms (single processor and multi-processor architectures) are defined to supply the default CFS application configuration. These configurations define the "maximum" CFS Application deployments such that any refined deployment is a subset of a reference platform. **1.2 Document Scope** The scope of this document is limited to the specification of requirements for the CFDP Software requirements. These include functional, performance, qualification, and design requirements. **1.3 Document Organization** This document is organized into three additional sections and several appendices. Section 2 gives the CF context. Section 3 documents the CF system design decisions and constraints. Section 4 contains the CF functional and performance requirements. Appendix A contains a list of abbreviations and acronyms used in this document. **1.4 Relevant Documents** **1.4.1 Parent Documents**		

None

1.4.2 Reference Documents

1. Operating System Abstraction Layer (OSAL) Library
2. Core Flight Executive (cFE) Application Developer's Guide   582-2007-001
3. cFE User's Guide
4. CCSDS 727.0-B-4 (CFDP Blue Book)
5. CFDP Engine User's Guide
6. CFS CF Design Review Version 2.0 (powerpoint slides)

2.0 CFS CF Application Context

The CFS CF Application provides the interface for transmitting and receiving files to/from the ground. CF utilizes CFDP as defined in the CCSDS 727.0-B-4 Blue Book.

To transfer files in the uplink or downlink direction, both a flight CFDP engine and a ground CFDP engine are required. The flight engine is contained in the CF application. The ground engine is typically integrated into the ground system.

The CFS CF context shows use of a complete CFS, presenting interfaces with other CFS applications. SCH is the CFS scheduler application that submits periodic housekeeping requests and Wake-up commands to CF. Ground commands come from a Command Ingest application (CI). Event messages and housekeeping packets are routed to the appropriate CFS output application, the Housekeeping (HK), Telemetry Output (TO), and/or Data Storage (DS) application. All accesses to the file system(s) are through the OS Abstraction Layer (OSAL) of the cFE.

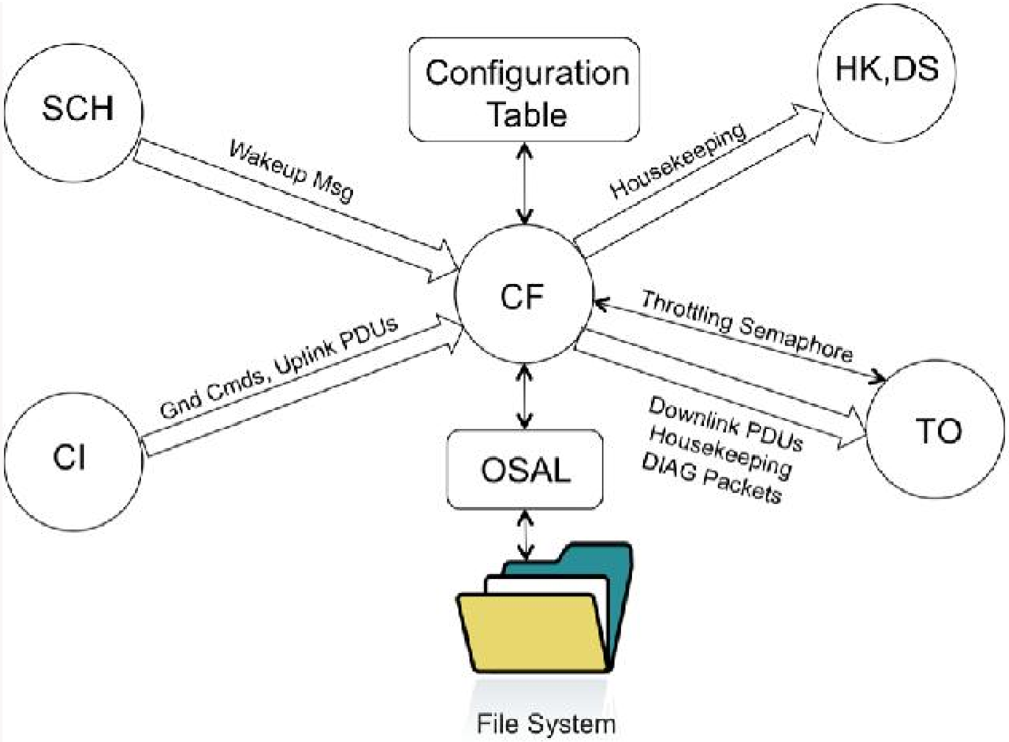


Figure 2.0 – CFS CF Context

The CF application has two major components, the CFDP flight engine and an outer layer interface to the engine. The CF application utilizes the CFDP Engine Developed by Code 583, SED Ground Systems Branch. The engine handles the CFDP protocol details and is capable of sending and receiving files simultaneously in Class 1 mode (unreliable) or Class 2 mode (Reliable). Each file transfer is broken down into a series of PDUs that are created by the engine. These PDUs contain file data or protocol messages. The outer layer passes the PDUs from the engine to the Software Bus (SB) for downlink transfers and from the SB to the engine for uplink transfers. The outer layer also provides a ground command interface and sends CF housekeeping data periodically.

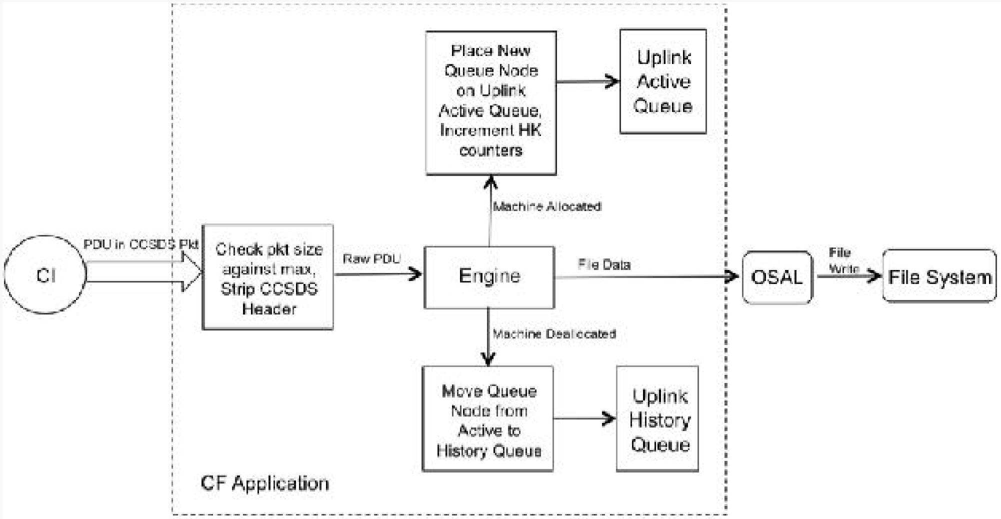

Place New Queue Node on Uplink Active Queue, Increment HK counters
Uplink Active Queue
Machine Allocated
CI
PDU in CCSDS Pkt
Check pkt size against max, Strip CCSDS Header
Raw PDU
Engine
File Data
OSAL
File Write
File System
Machine Deallocated
Move Queue Node from Active to History Queue
Uplink History Queue
CF Application

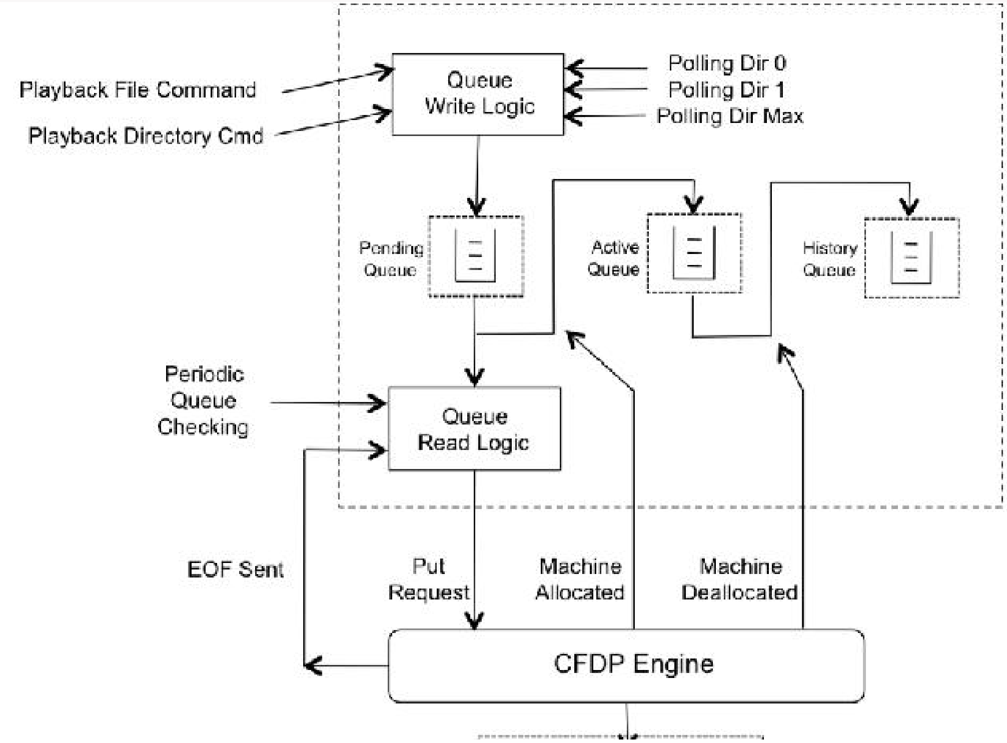
Figure 2.1 – CFS CF UPLINK PROCESSING

From the ground point of view, uplink transfers begin when the ground engine receives a put-request. A put-request is a request to the engine to begin outputting a file. In response to this put-request, the engine outputs a series of PDUs until the entire file has been transferred. From the CF application point of view, uplink transfers begin when the flight engine begins receiving the uplink PDUs. A command to the CF application to begin an uplink transfer is not needed and therefore does not exist.

For uplink file transfers, the CF outer layer receives PDUs that are enclosed in CCSDS packets from the Software Bus. The outer layer then strips off the CCSDS header and passes the raw PDU to the engine.

The outer layer does not need to look inside the PDUs. The outer layer maintains an uplink active queue and an uplink history queue. The queue nodes contain information about the transfer. Each node corresponds to one file transfer. When a transfer is active, the information regarding that transfer is held in the active queue. When the transfer is complete the queue node is read from the active queue and placed on the history queue. The CF application has a command to write the queue contents to a file so that it can be viewed on the ground.

The first PDU of a new transfer is a protocol message named meta-data and tells the engine the details of the transfer. In response to the meta-data PDU, the engine creates a temporary file with <PLATFORM_DEFINED, "/ram/cftmp" > directory/basename and leaves it open for writing. The engine will then receive a series of file-data PDUs. The engine writes the file data to the open file. At some point, the flight engine will receive an EOF PDU which tells the engine that all the file data should have been received. In the case of a Class 1 transfer, the engine would respond to the EOF PDU by closing and renaming the temporary file (to the destination path and filename) and performing a checksum test. In the case of a checksum failure, or if the temporary file cannot be renamed, an error event will be sent and the outer layer will count this as a failed transaction.



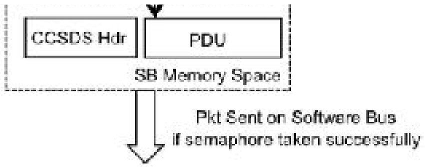


Figure 2.2 – CFS CF PlaybAck Processing

CF provides a maximum number <PLATFORM_DEFINED, 2> of playback channels. If the system has only one way for files to get to the ground, (such as via the Telemetry Output application) then only one channel is needed. If there is more than one path to the ground, then the CF application can be scaled to have more than one channel. Each channel contains a pending queue, active queue, history queue, throttling semaphore, polling directories, PDU buffer and separate message IDs for PDUs.

A simple playback example would start with the ground sending a CF command to playback a file. The outer layer receives this command from the software bus and stores the filename and other information in the pending queue (rather than giving the request to the engine immediately). The actual transfer does not begin until the flight engine receives a put-request from the outer layer. A put-request is a request to the engine to begin outputting a file.

The outer layer buffers the playback requests in the pending queue (one queue entry per file) so that the engine does not get inundated with transfer requests when a series of 'playback file' commands are received. The outer layer feeds the engine the transfer requests (or put-requests) one at a time. If dequeuing is enabled, the outer layer feeds to engine the transfer requests one at a time.

If there are no files in progress, the outer layer checks the pending queue periodically for playback requests. When a request is found on the pending queue, the outer layer immediately gives the engine a put request to start the transfer (if dequeing is enabled). Once the transfer is in progress, the pending queue is no longer checked periodically for transfer requests. When all the file data of the transaction has been sent, the outer layer gets an EOF-PDU-was-sent indication from the engine. At this time, the outer layer reads the pending queue and gives the engine a put request to start transferring the next file. Also when the outer layer receives a Machine-Allocated indication from the engine, the node representing the new transfer is moved from the pending queue to the active queue. A Machine-Allocated indication is the engines way of specifying that a new file-transfer state machine is created within the engine.

When a transfer is complete, the outer layer gets a Machine-Deallocated indication from the engine. This indication tells the outer layer that the engine has deallocated the internal, file-transfer state machine. In response to this indication, the outer layer moves the corresponding queue node from the active queue to the history queue.

Each playback channel defined in the configuration table has a maximum number <PLATFORM_DEFINED, 8> of polling directories. The user places files for downlink in these directories. CF will check the directories periodically, (at a table defined rate) and place all closed files found in the pending directories on the pending queue. The processing of the polling directories may be turned on or off by command.

The desired file order in the pending queue may vary each mission. It may be based on file age, file size, file name, priority or 'don't care' for maximum performance. The pending queue sorting algorithm is done in a callback routine to provide the most flexibility to each project. As a default, the CFS delivers a single callback routine that sorts the playback pending queue by priority. This callback routine supports up to 256 priority levels. See Appendix B for more information on CF callback routines.

During an active file playback, PDUs are being sent from the engine to the Software Bus. At most one PDU per engine cycle is sent to the Software Bus. This rate may be too fast for the downlink application (ex. Telemetry Output or TO). To accommodate this, CF allows the downlink application to control the flow of downlink PDUs. This is done through a counting semaphore that is 'given' by the downlink application and 'taken' by the CF application. If the engine has a PDU ready to go out, it first checks with the outer layer to see if it can be released. Based on the response from a semaphore 'take', the outer layer will hold off the engine or allow the engine to send the PDU.

As stated above the CF application has one table, the configuration table. This table allows a configurable number of playback channels. It is loaded during initialization and is not checked during run time for updates. It can be updated at anytime, but CF will find the updates only after an application restart. Many table parameters can be changed through command. The Configuration Table parameters include:

- Table Name
- Table Version
- Number of Engine Cycles Per Wake-up
- Ack Timeout Value
- Ack Limit Value
- Nak Timeout Value
- Nak Limit Value
- Inactivity Timeout Value
- Outgoing File Chunk Size
- Option to Save Incomplete Files or Not.
- Flight Entity Id
- Uplink History Queue Depth
- Downlink Channel Parameters (repeated for each channel)
    - Dequeue Enable
    - Downlink PDU MsgId
    - Pending Queue Depth
    - History Queue Depth
    - Channel Name
    - Handshake Semaphore Name
    - Polling Directory Parameters (repeated for each polling directory)
        - Enable State
        - Class
        - Priority
        - Source Path
        - Destination Path
        - Temporary Path
        - Preserve file
        - Peer Entity Id
- Uplink Channel Parameters (repeated for each channel)
    - Uplink PDU MsgId
    - Class 2 Uplink Response Channel

2.1 Assumptions

The following list summarizes the assumptions made by the CFS CF Application:

- cFE API and OSAL are being used .

- GSFC, Code 583 developed CFDP engine is used. This engine is capable of transmitting and receiving class 1(Unreliable) or class 2(Reliable) transfers. The engine duties include:

o Core or inner layer of the CF Application

o Implements the CFDP protocol

o Interprets and produces PDU's

o Reads blocks of file data from the file system for downlink

o Writes blocks of file data to the file system for uplink

3.0 Design Specifications

The CF Application's requirements and design are based on the results of meetings with the LRO FSW developers, MMS and GPM FSW teams, and LRO and GPM Operations teams.

The CF Application is designed to operate within the CFS architecture environment, which includes utilization of the cFE and OSAL. The design is based on the LRO CF application.

3.1 Design Constraints

The CF Application resides on the cFE and the OSAL. In addition, CF exists in the context of the CFS architecture

4.0 Subsystem Requirements

10116	CFS-700	The CFS shall deliver file data to the ground using the CFDP Class 1 and Class 2 services as specified in CCSDS 727.0-B-4 (Blue Book)		LRO
10118	CFS-701	The CFS shall receive file data from the ground using the CFDP, Class 1 and Class 2 services as specified in CCSDS 727.0-B-4 (Blue Book).		LRO
10120	CFS-702	The CFS shall format all information sent to the ground, related to CFDP file transfers, into CFDP Protocol Data Units (PDUs) compliant with CCSDS 727.0-B-4 (Blue Book).		GPM L4
10122	CFS-703	The CFS shall receive and interpret information, related to file transfers, from the ground formatted as CFDP Protocol Data Units (PDUs) compliant with CCSDS 727.0-B-4 (Blue Book).		
10124	CFS-704	The CFS shall send a single file or all files in a specified directory to the CFDP playback queue on command.		
10126	CFS-705	The CFS shall allow CFDP protocol timers and counters to be changed by command.		

9884		**5.0 Detailed Requirements** **5.1 Basic Requirements** The following requirements are basic requirements of the CF application. Some of them are included here to avoid repeating these requirements for each applicable requirement.		
10142	CF1000	Upon receipt of a No-Op command, CF shall increment the CF Valid Command Counter and generate an event message.	Debug command to verify application is alive.	LRO
10144	CF1001	Upon receipt of a Reset Counters command, CF shall reset the command-specified housekeeping variables to a value of zero.	Important for testing and on-orbit flight operations in order to start with a “clean slate”. Values to be reset will be specified via command-specified bit mask.	LRO
10146	CF1002	For all CF commands, if the length contained in the message header is not equal to the expected length, CF shall reject the command and issue an event message.	Basic command verification in the event of SEU or memory corruption.	LRO
10148	CF1003	If CF accepts any command as valid, CF shall execute the command, increment the CF Valid Command Counter and issue an event message.	Operators require feedback on command execution.	LRO
10150	CF1004	If CF rejects any command, CF shall abort the command execution, increment the CF Command Rejected Counter and issue an error event message.	Operators require feedback on command execution.	LRO
10152		**5.2 File Receiving**		
10154	CF2000	The CF application shall receive files from the ground using CFDP Unacknowledged Mode (Class 1 Service) or CFDP Acknowledged Mode (Class 2 Service).		LRO
10156	CF2001	The CF application shall extract uplinked CFDP PDUs from SB messages.		LRO
10158	CF2002	The CF Application shall extract file data from File Data PDUs and construct a complete file in the meta-data specified directory identical to the original transmitted file.		LRO
10160	CF2002.1	CF shall cancel the transaction and issue an event message if the engine detects that a “Fault” has occurred.		LRO
10162	CF2002.1.2	The following faults shall be detected: a. Positive Ack Limit Reached d. Filestore Rejection e. File Checksum Failure f. File Size Error g. NAK Limit Reached h. Inactivity Detected j. Suspend Request Received		

10164	CF2003	DELETED: Upon receiving a metadata PDU that specifies a restricted destination directory, the CF Application shall issue an error event and cancel the transaction.	Restricted directories are configuration parameters	
10166	CF2004	DELETED: Upon receiving a file that exceeds the <PLATFORM_DEFINED> maximum file size, CF shall issue an error event and cancel the transaction.	Data that has been received is left in the temporary directory.	
10168		**5.3 FIle Playback**		
10170	CF3000	The CF Application shall, upon command, playback a command-specified file using CFDP Unacknowledged Mode (Class 1 Service) or CFDP Acknowledged Mode (Class 2 Service) using a command-specified playback priority, playback channel number, and peer entity id.		LRO
10172	CF3000.1	CF shall issue an error event message and reject a playback file command if command-specified file is currently open.		LRO
10174	CF3000.2	CF shall issue an error event message if the command-specified file is currently being played back.		LRO
10176	CF3000.3	CF shall issue an error event message if the command-specified file is not found		LRO
10178	CF3000.4	The CF application shall issue an error event message and reject a playback file command if the playback pending queue is full		
10180	CF3000.5	The CF application shall issue an error event message and reject a playback file command if the command-specified playback channel number is not valid		
10182	CF3000.6	The CF application shall issue an error event message and reject a playback file command if the command-specified class number is not valid		
10184	CF3000.7	DELETED: The CF application shall issue an error event message and reject a playback file command if the command-specified priority is not valid.		
10186	CF3001	The CF Application shall, upon command, playback all files from the command-specified directory, excluding subdirectories, using a command-specified playback priority, playback channel, and peer entity id.		LRO
10188	CF3001.1	The CF Application shall skip playback of files that are open.		LRO
10190	CF3001.2	DELETED: The CF Application shall skip playback of files that are already on the playback queue.		
10192	CF3001.3	The CF application shall issue an error event message if the command-specified directory is not found.		
10194	CF3001.4	DELETED: The CF application shall issue an error event message and reject a playback directory command if the command-specified directory is a restricted directory.		
10196	CF3001.5	The CF application shall issue an error event message and reject a playback directory command if the playback pending queue is full.		
10198	CF3001.6	The CF application shall issue an error event message and reject a playback directory command if the command-specified playback channel number is not valid		
10200	CF3001.7	The CF application shall issue an error event message and reject a playback directory command if the command-specified class number is not valid		
10202	CF3001.8	DELETED: The CF application shall issue and error event message and reject a playback directory command if the command-specified priority of not valid.		
10204	CF3002	DELETED: The CF Application shall, upon command, cancel the playback of a command-specified file and issue an event message indicating the playback was canceled.		LRO
10206	CF3002.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified file is not currently being played backed.		
10208	CF3003	DELETED: The CF Application shall, upon command, cancel the playback of all transactions in progress and issue an event message indicating the playback was cancelled.		
10210	CF3003.1	DELETED: The CF application shall issue an error event message if there are no transactions in progress.		
10212	CF3004	DELETED: The CF Application shall delete a file that has been successfully played back using Class 2 Service.	See Appendix B for more details	LRO
10214	CF3005	The CF Application shall encapsulate all playback CFDP PDUs in SB messages.		LRO
10216	CF3005.1	The CF Application shall assign the Message ID given in the configuration table.		
10218	CF3006	CF shall allow another application to control the PDU output rate.	This is to ensure that CFDP does not overload the downlink application.	
10220	CF3006.1	DELETED: If CF handshake is enabled, the CF application shall not send downlink packets to the software bus until it receives acknowledgement from the downlink application.		

10222	CF3007	The CF application shall, if there is a series of files to be played back, wait until the engine indicates "EOF Sent" on the file currently being played back before starting playback for the next file in the playback pending queue on the corresponding playback channel.	The code to start the next file will be invoked in the 'EOF Sent' callback routine.	LRO
10224	CF3008	If there is no playback in progress, CF shall check the playback pending queue for a file at the Table_Defined rate		
10226	CF3009	The CF application shall provide a call-back routine that executes each time a file or group of files is placed on the playback pending queue.	See Appendix B for more details	
10228	CF3010	DELETED: The CF application shall provide a command to suspend the command-specified playback transaction in progress.	Suspend is used to pause transactions during contact. Useful during safe-hold to ensure that only engineering data is sent	
10230	CF3010.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified playback transaction is not in progress.		
10232	CF3011	DELETED: The CF application shall provide a command to suspend the playback of all transactions in progress.		
10234	CF3011.1	DELETED: The CF application shall issue an error event message and reject the command if no transactions are in progress.		
10236	CF3012	DELETED: The CF application shall provide a command to resume the command-specified paused playback transaction.	Resume starts suspended transactions.	
10238	CF3012.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified playback transaction is not currently suspended.		
10240	CF3013	DELETED: The CF application shall provide a command to resume the playback of all paused transactions.		
10242	CF3013.1	DELETED: The CF application shall issue an error event message and reject the command if there are no suspended transactions.		
10244	CF3014	DELETED: The CF application shall provide a command to purge a command-specified playback pending queue	Playback already in progress will be completed.	
10246	CF3014.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified playback pending queue is not defined.		
10248	CF3015	DELETED: The CF application shall provide a command to dump the command-specified playback queue to the command-specified file		
10250	CF3015.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified playback queue is not defined.		
10252	CF3016	DELETED: The CF application shall provide a command to dump the command-specified playback list to the command-specified file.		
10254	CF3016.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified playback list is not defined.		
10256	CF3017	DELETED: The CF application shall provide a command to delete a command-specified file on the command-specified playback pending queue		
10258	CF3017.1	DELETED: The CF application shall issue an error event message and reject the command if the command-specified file is not found.		
10260	CF3017.2	DELETED: The CF application shall issue an error event message and reject the command if the command-specified playback pending queue is not defined.		
10262		**5.4 Queue Management for File Playback**		
10264	CF4000	The CF application shall allow a table-defined number of playback channels.	Allows missions to tune the output to what is most optimal to meet their requirements	
10266	CF4000.1	CF utilizes a Configuration Table to define these playback channels.	Tables are used so that playback outputs can be configured for a mission.	

10268	CF4000.1.1	The Configuration Table provides following configuration items: A. Playback channel parameters (repeated for each channel) a. Dequeue Enable c. Downlink PDU MsgId d. Pending Queue Depth e. History Queue Depth g. Channel Name h. Handshake Semaphore Name i. Polling Directory Parameters (repeated for each polling directory) 1. Enable State 2. Class 3. Priority 4. Source Path 5. Destination Path 7. Preserve file 8. Peer Entity Id B. Input channel parameters (repeated for each channel) a. Input PDU MsgId b. Class 2 Uplink Response Channel		
10270	CF4000.2	The CF application shall create a dedicated playback pending queue for each playback channel.		
10272	CF4000.3	The CF application shall allow a unique message ID for the output PDU on each playback channel.		
10274	CF4001	The CF application shall provide a command to enable or disable poll directory processing.		
10276	CF4001.1	Polling directories will be polled for files at the table defined rate.		
10278	CF4001.2	DELETED: The CF application shall provide a command to enable or disable poll directory processing.	Gives the ground the ability to run with or without a Hot Directory.	
10280	CF4001.2.1	The CF application shall place all files found in the polling directory on the corresponding playback pending queue at the specified priority level.		
10282		**5.5 Miscellaneous**		
10284	CF5000	The CF application shall provide a command to freeze all transactions in progress.	Freeze command is used to pause everything when not in contact	
10286	CF5001	The CF application shall provide a command to thaw all frozen transactions.	Thaw command is used to resume everything when contact is resumed	
10288	CF5002	The CF application shall support the following CFDP protocol timer configurations via commands: a. Modify the following protocol timer settings: 1.1 Ack Timer Value (seconds) 1.2 NAK Timer Value (seconds) 1.3 Inactivity Timeout Value (seconds) b. Modify the following protocol counter settings 2.1 Max Number of Ack Timeouts 2.2 Max Number of Nak Timeouts		
10290	CF5002.1	The CF application shall allow default timer/counter values to be set via table parameters		
10292	CF5003	Upon command CF shall issue an event message containing the current value of the command-specified timer/counter setting .		
10294	CF5003.1	The CF application shall issue an error event message and reject the command if the command-specified configuration parameter is not defined.		
10296	CF5004	Upon command CF shall send all configuration settings in a telemetry message to the ground.	Allows ground to easily determine CFDP configuration.	
10298	CF5005	The CF application shall provide a command to cancel a command-specified transaction.		
10300	CF5005.1	The CF application shall issue an error event message and reject the command if the command-specified transaction is currently not in progress.		

10302	CF5006	The CF application shall provide a command to abandon a command-specified transaction.		
10304	CF5006.1	The CF application shall issue an error event message and reject the command if the command-specified transaction is not currently in progress.		
10306	CF5007	The CF application shall provide a command to abandon all transactions.		
10308	CF5007.1	The CF application shall issue an error event message and reject the command if there are currently no transactions in progress.		
10310	CF5008	The CF application shall provide a command to send a diagnostics packet for a command-specified transaction		
10312	CF5008.1	The CF application shall issue an error event message and reject the command if the command-specified transaction is not found in the pending, active or history queue.		
10314	CF5009	The CF application shall provide a command to write all active transactions to a file.		
10316	CF5009.1	DELETED: The CF application shall issue an error event message and reject the command if there are currently no active transactions.		
10318	CF5010	DELETED: The CF application shall provide a command to write the history queue entries for the command specified history queue		
11589	CF5010.1	DELETED: The CF application shall issue an error event message and reject the command if the command specified playback channel is invalid.		
10320	CF5011	DELETED: The CF application shall allow up to 4 write-restricted directories.	Allows mission to restrict the use of non-volatile memory.	
10322	CF5012	The CF application shall support a maximum uplink PDU size of <PLATFORM_DEFINED> bytes.	This is a global setting that applies to uplink and downlink PDUs for all output strings.	
10324	CF5013	The CF application shall allow a maximum of <PLATFORM_DEFINED> simultaneous transactions to occur at once.		
11591	CF5014	The CF application shall support a maximum downlink PDU size of <PLATFORM_DEFINED> bytes.		
11593	CF5015	The CF Application shall, upon command, cancel all transactions in progress.		
11595	CF5015.1	The CF application shall issue an error event message if there are no transactions in progress.		
11597	CF5016	The CF application shall provide a command to suspend the command-specified transaction in progress.		
11599	CF5016.1	The CF application shall issue an error event message and reject the command if the command-specified transaction is not in progress.		
11601	CF5017	The CF application shall provide a command to suspend all transactions in progress.		
11603	CF5017.1	The CF application shall issue an error event message and reject the command if no transactions are in progress.		
11605	CF5018	The CF application shall provide a command to resume the command-specified paused transaction.		
11607	CF5018.1	The CF application shall issue an error event message and reject the command if the command-specified transaction is not currently suspended.		
11609	CF5019	The CF application shall provide a command to resume all paused transactions.		
11611	CF5019.1	The CF application shall issue an error event message and reject the command if there are no suspended transactions.		
11613	CF5020	The CF application shall provide a command to purge a command-specified playback pending or history queue		
11615	CF5020.1	The CF application shall issue an error event message and reject the command if the command-specified queue is not defined.		
11617	CF5021	The CF application shall provide a command to write the command-specified queue contents to the command-specified file.		
11619	CF5021.1	The CF application shall issue an error event message and reject the command if the command-specified queue is not defined.		
11621	CF5022	The CF application shall provide a command to dequeue a command-specified file on the playback pending or history queue		
11623	CF5022.1	The CF application shall issue an error event message and reject the command if the command-specified file is not found.		
11625	CF5022.2	DELETED: The CF application shall issue an error event message and reject the command if the command-specified queue is not defined.		
10326		**5.6 Status Reporting**		

10328	CF6000	CF shall generate a housekeeping message containing the following: a. Ground Command Error Counter b. Ground Command Counter c. Playback Pending Queue State d. Files on Playback Pending Queue e. Frozen Status f. Transactions In Progress g. Completed Transactions h. Failed Transactions i. Uplink PDUs Received j. Uplink Transactions In Progress k. Uplink Transactions Succeeded l. Uplink Transactions Failed n. Filename of Last Successful Uplink o. Downlink Transactions In Progress p. Downlink Transactions Succeeded q. Downlink Transactions Failed s. Downlink PDUs Sent t. Ack Timer Limit Faults u. Filestore Rejection Faults v. Checksum Failure Faults w. Filesize Error Faults x. NAK Limit Faults y. Inactivity Timer Limit Faults z. Cancel Request Faults aa. Memory pool allocation remaining bb. Number of transactions abandoned	Housekeeping telemetry to indicate basic CF status.	LRO, SDO
10330		**5.7 Initialization Requirements** CF has no requirements to preserve anything across a CF application or processor reset. A CF application or processor reset, therefore looks like a cFE Power-on reset. Any initialization required for any device would need to be done outside of CF.		
10332	CF7000	Upon initialization of the CF Application shall initialize the following data to Zero (or the value specified for the item below): a. Ground Command Error Counter b. Ground Command Counter c. Playback Pending Queue State - enabled d. Files on Playback Pending Queue e. Frozen Status - thawed f. Transactions In Progress g. Completed Transactions h. Failed Transactions i. Uplink PDUs Received j. Uplink Transactions In Progress k. Uplink Transactions Succeeded l. Uplink Transactions Failed n. Filename of Last Successful Uplink o. Downlink Transactions In Progress p. Downlink Transactions Succeeded q. Downlink Transactions Failed s. Downlink PDUs Sent t. Ack Timer Limit Faults u. Filestore Rejection Faults v. Checksum Failure Faults w. Filesize Error Faults x. NAK Limit Faults y. Inactivity Timer Limit Faults z. Cancel Request Faults bb. Number of transactions abandoned	.	Derived
10334	CF7001	Upon any Initialization, CF shall load the Configuration Table.	Need default values	
10336	CF7001.1	If the Configuration Table fails validation, CF shall issue an event message and exit.	Because the table is a configuration type table, CF can proceed only if all table entries are valid.	

10338	CF7001.1.1	CF shall validate the following values in the Configuration table: a. Outgoing File Chunk Size b. Flight and Ground Entity Ids c. Message IDs d. CFDP Class 1 or 2		
9886		**Appendix A Acronyms and Abbreviations** This appendix contains the list of abbreviations and acronyms for the CFS CF Application used in this document. These abbreviations and acronyms adhere to FSB standard usage. · **Ack** – Acknowledgement · **API** – Application Program Interface · **CCSDS** - Consultative Committee for Space Data Systems · **CF** – CFDP Application · **CFDP –** CCSDS File Delivery Protocol · **cFE** - Core Flight Executive · **CFS** – Core Flight System · **CI** – Command Ingest · **DS** – Data Storage · **EOF** – End of Frame · **FSB** - Flight Software Branch · **FSRL** – Flight Software Branch Reuse Library · **FSW** – Flight Software · **GPM** – Global Precipitation Mesurement · **HK** - Housekeeping · **ID** - Identifier · **LRO** – Lunar Reconnaissance Orbiter · **MMS** – Magnetosphere Multiscale · **MsgId** – Message Identifier · **NAK** – Negative Acknowledgement · **OSAL** – Operating System Abstraction Layer · **PDU** – Protocol Data Units · **SB** – Software Bus		

- **SCH** - Scheduler
- **SED** – Software Engineering Division
- **TO** – Telemetry Output

Appendix B CF Call Back Routines

The CF application utilizes several callback routines to allow missions to easily customize the processing of several different CF task. Callback routines are used for playback priority sorting, playback hardware assist enable and post Class 2 playback file handling. The default functionality of these callback routines is described below:

Default Playback Priority Sorting Routine

The desired file order in the playback pending queue may vary each mission. It may be based on file age, file size, file name, priority or 'don't care' for maximum performance; therefore the playback pending queue sorting algorithm is done in a callback routine to provide the most flexibility to each project. As a default, the CFS delivers a single callback routine that sorts the playback pending queue by priority.

The default sorting routine delivered with the CF application queues playback files according to their assigned priority. CF supports a maximum of 256 level of priority and the priority is set via the playback command.

Default EOF Sent Message Response

The response to the EOF Sent message may vary with each mission. The default response is to start the transmission of the next file on playback queue.