Core Flight System (CFS)

CFDP (CF) Flight Software User's Guide

Document Date: August 26, 2011 Version 1.0

Author:

_____ _____
Robert McGraw/CFS Flight Software Developer Date

Approvals:

_____ _____
Raymond Whitley/CFS Flight Software Product Development Lead Date

_____ _____
Walter Moleski/CFS Flight Software Test Lead Date

Revision History

Revision Number	Release Date	Changes to Prior Revision
1.0	08/26/2011	Initial Release – Command and Telemetry definitions as well as events, tie this document to CF version 2.2.x.

Table of Contents

1 Introduction

1.1 Scope

This document provides a complete specification for the configuration, commands and telemetry associated with the CCSDS File Delivery Protocol (CF) flight software application version 2.2.0. The document is intended primarily for users of the software (operators, testers, and maintainers).

1.2 Applicable Documents

Document ID	Document Title
582-2007-007	CFS FSW Development Standards Document
CCSDS 727.0-B-4	CCSDS CFDP Blue Book
582-2009-006	CFS CF Requirements Document V1.1

1.3 Acronyms

Acronym	Description
API	Application Programming Interface
APID	CCSDS Application ID, subset of Message ID
App	Flight Software Application
Asist	Advanced Spacecraft Integration and System Test
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
CF	CFdp Flight Software Application
C&DH	Command and Data Handling
CM	Configuration Management
CFE	Core Flight Executive
CFS	Core Flight System
Cmd	Command
EOF	End Of File
ES	cFE Executive Services
FOPS	Flight Operations
FSW	Flight Software
GSFC	Goddard Space Flight Center
HK	Housekeeping
ID	Identifier/Identification
Mbps	Mega bits per second
MBytes	Mega bytes
MD	Meta Data
MIB	Message Information Base
OS	Operating System
OSAL	Operating System Abstraction Layer
PDU	Protocol Data Unit

Acronym	**Description**
Pkts	Packets
SB	Software Bus Service
SC	Stored Commands application
SW, S/W	Software
TBD	To Be Determined
TBL	Table
TCP	Transmission Control Protocol
TLM	Telemetry
TO	Telemetry Output application
UDP	User Datagram Protocol
VC	Virtual Channel

2 CF Application Function

2.1 Overview

The CF application is a flight software application used for transmitting and receiving files. It is a CFS application and interfaces to the Core Flight Executive (cFE).

To transfer files using CFDP, the CF application must communicate with a CFDP compliant peer. CF may be configured to have any number of peers. The ASIST and ITOS ground system contains a compliant peer that is used for flight to ground (and ground to flight) transfers.

CF sends and receives file information and file-data in Protocol Data Units (PDUs) that are compliant with the CFDP standard protocol defined in the CCSDS 727.0-B-4 Blue Book. The PDUs are transferred to and from the CF application via CCSDS packets on the software bus.

There are features listed in the CFDP standard that are not applicable to flight software and are therefore not supported by this version of CF. See the constraints section in the Appendix for more information.

2.2 Terminology

2.2.1 Engine

The CF application has a single internal core called the 'engine'. The engine is capable of transmitting and receiving a configurable number of transactions simultaneously. The engine builds outgoing Protocol Data Units (PDUs) and interprets the incoming PDUs. It handles all details regarding the CFDP standard protocol defined in the CCSDS 727.0-B-4 Blue Book. The engine processes the file transactions when it is 'cycled'. The engine is cycled a configurable number (NumEngCyclesPerWakeup, table param) of times each time it receives a Wake-up command. At most one PDU will be sent on a single engine cycle. Typically, the peer node also has an engine. When CF is transferring files to and from the ground, the peer is typically the ground engine. When faults and timeouts occur, it is important to indicate which node (flight engine vs. ground engine) detected the event.

2.2.2 Entity ID

Every CFDP engine has an associated entity ID. An entity ID is similar to an abbreviated IP Address. Instead of the 4 byte dotted decimal format that IP Addresses use, the CF application requires that CF and all peers use a 2 byte dotted decimal format such as 0.24. The CFDP standard allows an entity id length to be one to four bytes, but CF will accept only an entity id length of two bytes. The CF configuration table will not pass validation if the entity ID length is other than two bytes. See section on Troubleshooting (#7) for symptoms related to Peer entity IDs that are not formatted properly. The highest possible entity ID would be 255.255 and the lowest would be 0.0. See section named CF Configuration Table Parameters in Appendix A for the entity IDs.

2.2.3 Transaction Number

The transaction number is a four byte integer. The peer that sends the file (source entity) assigns the transaction number. The CFDP standard allows the length of a transaction number to be between one and four bytes. The CF application allows only a length of four bytes for transaction numbers.
The first outgoing transaction after a reset gets a value of one. The same transaction number may be assigned to incoming transactions. To distinguish between incoming transaction number five for example, and outgoing transaction number five, the Source Entity ID is prefixed to the transaction number, creating a 'Transaction ID' (defined below).

2.2.4 Transaction ID

A file transaction is created for each incoming file received and each outgoing file sent. Each file transaction has a unique identifier formatted as 'SourceEntityID_TransactionNumber'. For example, if the flight entity ID was set to 0.24, the third file downloaded after a flight reset will have a transaction ID of '0.24_3'. If the ground CFDP entity ID set to '0.23'. The first file uplinked after a ground engine reset will have a transaction ID of '0.23_1'. See Transaction Number defined above for more detail.

2.2.5 Transaction Class

All transfers are sent and received in one of two modes, class 1 or class 2. The CF application is capable of sending and receiving in class 1 and class 2. Class 1 transfers are similar to UDP in that they send the data once and expect no feedback from the peer. Class 2 transfers are more reliable and attempt to fill in data that may have been dropped on the first attempt. Class 2 transfers are analogous to TCP.

2.3 CFDP Class 2 Nominal Transfer Protocol Messages

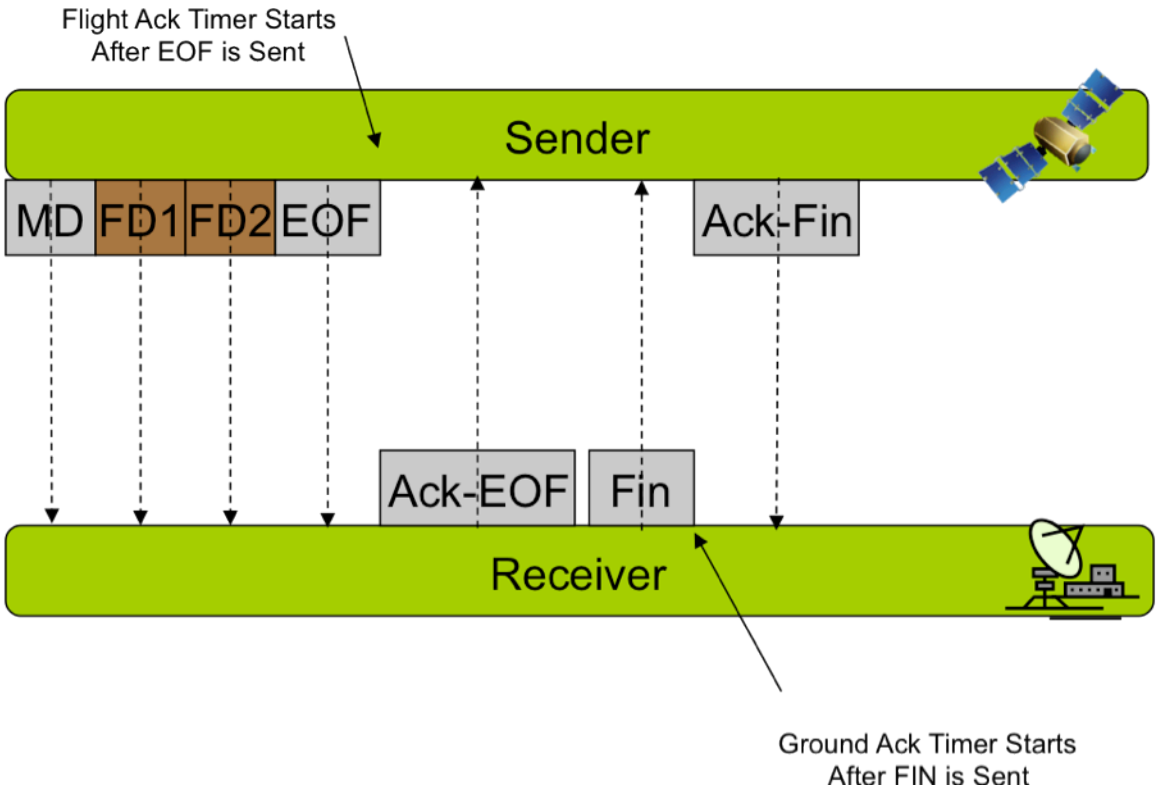

Figure 2-3. CFDP Protocol Messages – Nominal Transfer

2.4 CFDP Class 2 Protocol Messages – Missing Data

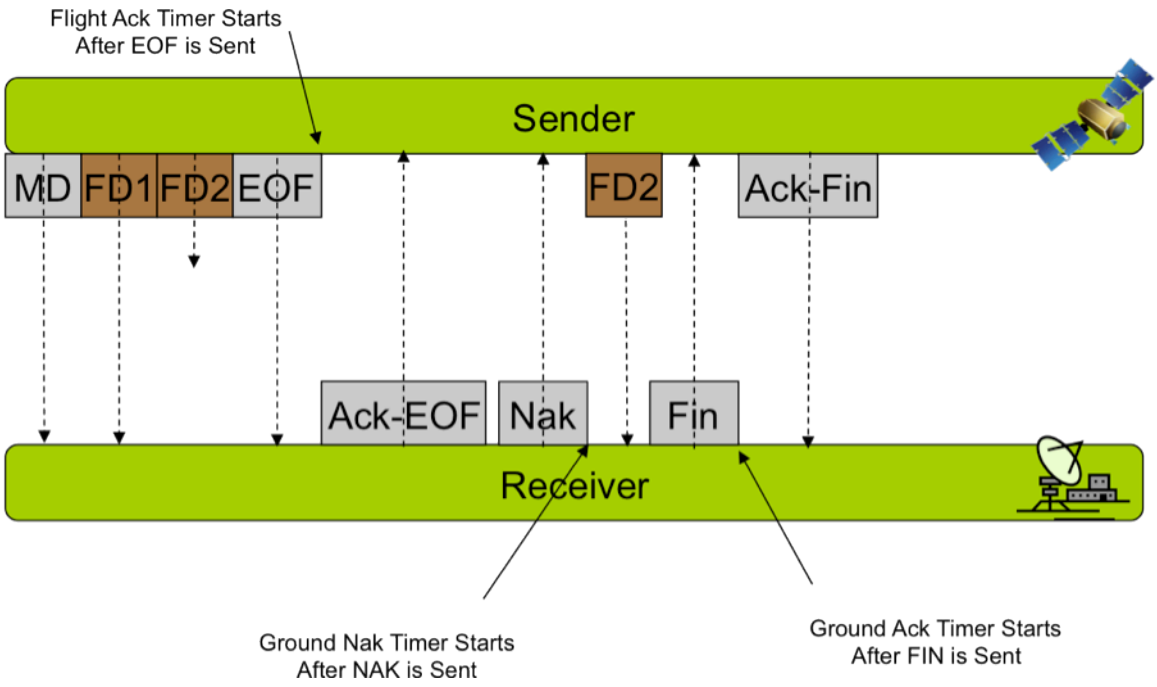

Figure 2-4. CFDP Protocol Messages – Missing Data

3 CF Software Design

3.1 CF Design Overview

The CF application is a flight software application that is used for transmitting and receiving files. It is a CFS application and interfaces to the Core Flight Executive (cFE).

CF is a configurable application that designed to be used on a wide range of flight missions. CF obtains its initial configuration through a configuration table and its platform and mission configuration files. The table contains default configuration settings and is loaded during CF initialization. The platform and mission configuration files are compile-time configuration parameters.

CF is an event driven, single threaded application that wakes up when one of the following four messages are received on its software bus pipe: Ground command, Housekeeping Request command, Incoming PDU or the Wake-up command. The Wake-up command is sent by the scheduler application and tells CF to do file transaction processing. The amount of file-transaction processing that is executed when this command is received, is configurable through the table parameter engine-cycles per wake-up.

For simplicity, the examples throughout this document refer to a typical operational scenario whereby the peer to the CF application is located on the ground. The CF application knows only of incoming file transactions and outgoing file transactions. The terms uplink, downlink and playback are often used, but only apply when the peer is located on the ground. CF may be configured to have more than one peer.

3.2 CF Context Diagram

The CF application interfaces with the files system through the OS Abstraction Layer (OSAL). CF receives commands from the Scheduler (SCH) by way of the cFE Software Bus. All outgoing PDUs and Telemetry Packets are sent on the Software Bus. CF communicates with TO for output PDU throttling through an OSAL counting semaphore.

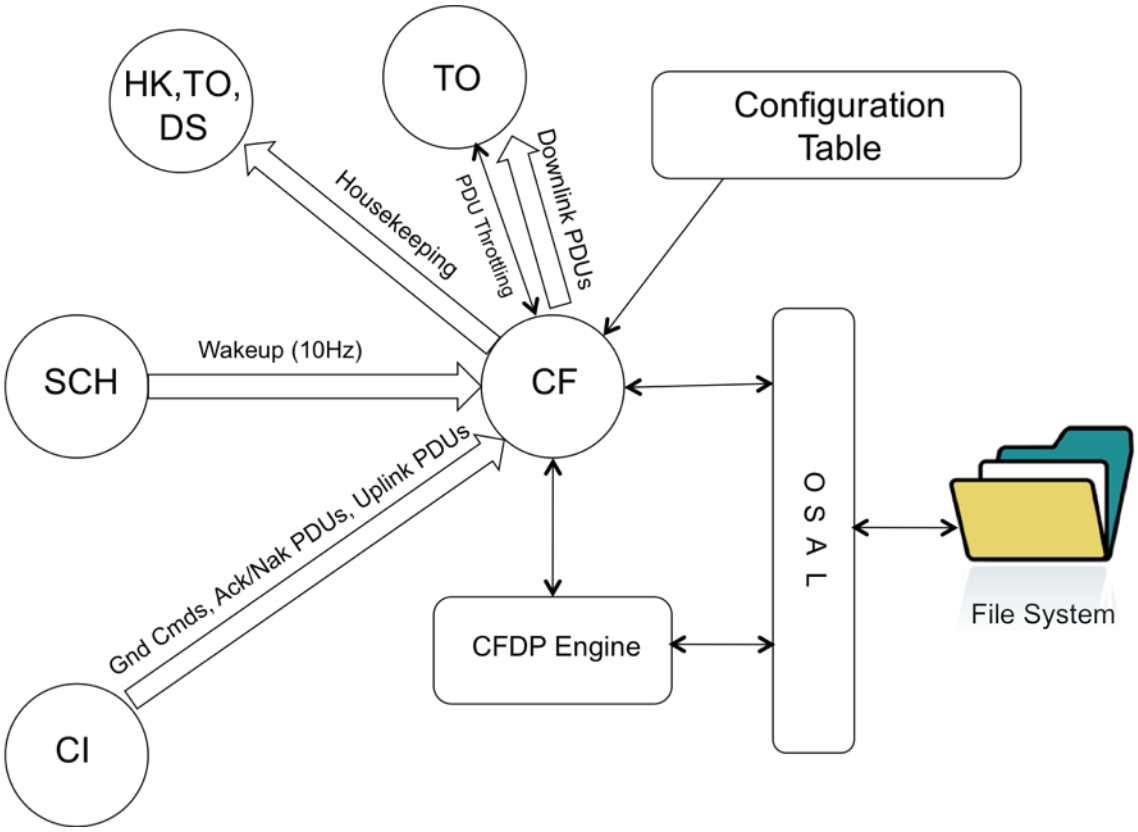

Figure 3-2. Context Diagram

3.3 CF Flow Control

The majority of the CF Software has a fairly standard task design: Following reset, the application is initialized, and then pends on the software bus.

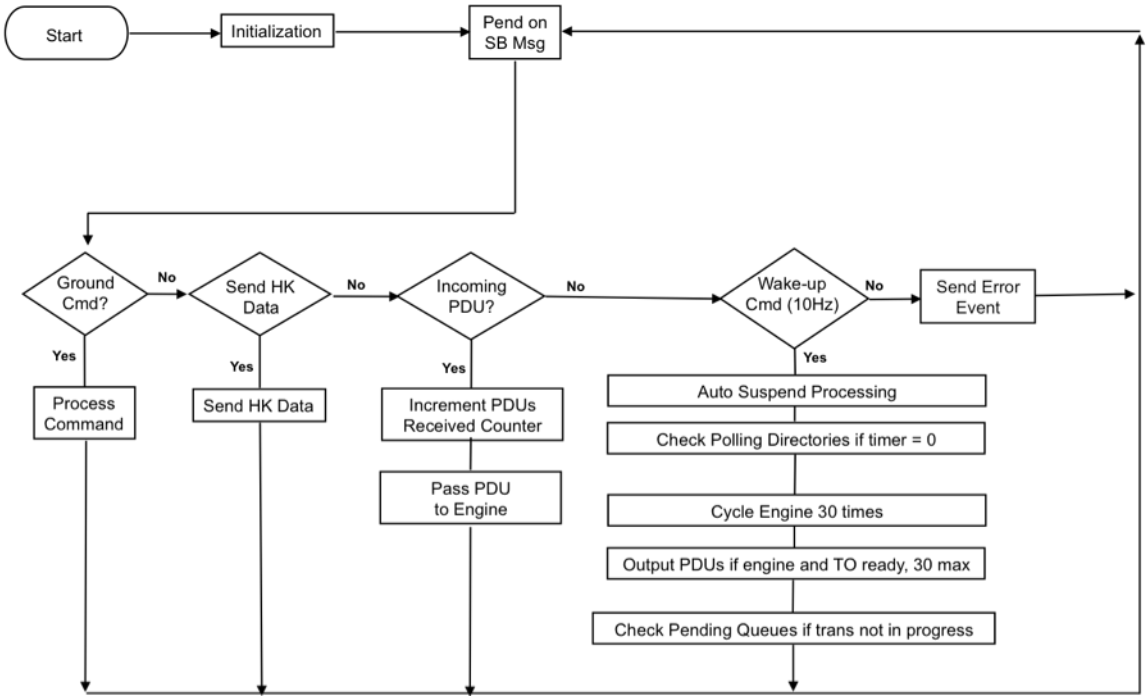

Figure 3-3. CF Flow Control

3.4 CF Detailed Design

3.4.1 Initialization

No action is required by the ground to initialize the CF software. During initialization CF reads its configuration table file and applies the default settings to the software. After initialization is complete, CF will send an event to indicate that initialization is complete. This event will display the version number of the application. If an error occurs during initialization, (for example, if the table file cannot be found or a table validation error occurs), the application will send an event indicating the failure before the application is terminated.

3.4.2 Power-On/Processor Resets

The CF Application initializes and executes in the same way after a power-on or processor reset. Following Power-On or Processor Reset, CF clears all telemetry counters. The engine is also reset and initializes its counters. The CF application does not use the Critical Data Storage area to save data across a reset.

3.4.3 Initiating A File Transaction

A transaction always begins with a request to send a file at the source entity (i.e. where the file is located). For example, uplink transactions begin with a request to send a file at the ground engine and downlink transactions begin when CF receives a playback command. Polling directory processing is also used to initiate a downlink transaction. Note that there is no ground command telling CF to 'get' a file. The CFDP protocol does not support the concept of a 'get' request. The first indication to CF that an uplink transaction has started, is the receipt of the first PDU sent by the ground.

3.4.4 CF Queue Entries

The CF application creates a queue entry for each transaction. The queue entries contain information about the transaction such as filename and path, priority, class, channel, peer entity ID. The queue entries are moved from queue-to-queue as their state changes. For example, when a transaction is complete, the queue entry for that transaction is moved from the active queue to the history queue. A single entry may be removed from any queue by way of the dequeue command.

3.4.5 CF Queues

The CF application has pending queues, active queues and history queues. The queues contain entries described in the previous section. The full contents of any queue can be viewed on command. The depth of the history queues and the pending queues are defined in the CF Configuration table. The depth of the active queues are limited only by the platform configuration parameter CF_MAX_SIMULATANEOUS_TRANSACTIONS. The history queues and pending queues may be cleared by way of the purge queue command.

3.4.5.1 Outgoing Transaction Queues

For downlink (or outgoing) transactions there are three queues per output channel, a sorted pending queue, an active queue and a history queue. All queues hold entries that are described in the 'Queue Entries' section of this document. When a request to downlink a file is received, a queue entry is allocated, populated and inserted on the pending queue. When the transaction begins, the queue entry is moved from the pending queue to the active queue. After the transaction completes, (whether successful or not) the queue entry is moved from the active queue to the history queue. The history queue has a fixed depth, defined by the user in the CF configuration table. If a transaction needs to be added to a history queue that is full, the oldest history queue entry is deleted to make room for the new entry. The deleted queue entry is then lost forever and its memory is returned to the heap.

3.4.5.2 Pending Queue Sorting Algorithm

Every output channel has a sorted pending queue. This queue holds queue entries for files that are waiting to be sent. The depth of the pending queue is specified in the CF configuration table. The pending queue is sorted by priority. Higher priority files are placed ahead of lower priority files on this queue. The user specifies the priority of a file as a parameter in a playback-file or playback-directory command. In the case of polling directory processing, each polling directory is assigned a priority in the CF configuration table. The priority values range from zero (being the highest) to 255. If there are multiple files on the pending queue with the same priority, the order within the priority is dictated by the order in which the files were queued. If a group of files with the same priority are added (as with a playback directory cmd or poll processing that found more than 1 file to be queued) then the files are placed on the queue within the priority, in alphabetical order.

3.4.5.3 Incoming Transaction Queues

For uplink (or incoming) transactions there are two queues, an active queue and a history queue. Uplink transactions do not have a pending queue as with outgoing transactions. When an uplink transaction begins, a queue entry is added to the active queue. When the transaction completes, the queue entry is moved from the active queue to the history queue. The user specifies the history queue depth in the CF configuration table. When the history queue is full and an active transaction finishes, the oldest entry on the history queue is deleted to make room for the new entry. The deleted entry is then lost forever and its memory is returned to the heap.

3.4.6 CF Incoming File Transactions

When files are transferred in the uplink direction, the ground peer receives the initial 'put' request to send the file. In this request the user specifies the class (1 or 2), the name of the file to send, the entity ID of the peer and the destination path and filename. This request is typically entered at the stol prompt on Asist and may look like this:

> cfdp_dir "put –class1 /s/opr/accounts/cfs/image/file1.txt 0.24 /ram/file1.txt"
> See section named Ground Engine Commanding and Status for more examples.

This action causes the ground peer to send a series of PDUs that are placed in CCSDS packets and routed to the CF application. The CF application does not get a request to receive a file.

The CF application is capable of receiving files in class 1 or class 2 mode on a per-file basis. The class mode is specified in the request and embedded in the PDUs.

When a file is uploaded to the spacecraft in class 2 mode, the CF app must acknowledge the receipt of the file by sending an Ack-EOF PDU to the ground. This response must be sent on a specified output channel (output channels are described later). The user defines the channel number for this response in the configuration table.

The CF application keeps a list of all incoming transactions in its internal queues. See section named CF Queues for more information.

3.4.7 CF Outgoing File Transactions

All outgoing file transactions are initiated by the CF application in response to a playback file command, a playback directory command or a file found in a polling directory. All outgoing file transactions are inserted into a pending queue before they are actually sent. The CF application reads the pending queue (if reading is enabled) and starts the next transaction immediately after the data from the previous file has been sent. This process of queuing files and sending them sequentially, prevents the engine from being inundated when multiple files need to be sent. Once the transaction begins, the queue entry is moved to the outgoing active queue and then to the outgoing history queue when it's complete. The engine processes the outgoing file transactions when it is 'cycled'. The number of engine cycles per wake up command is defined in the table. At most one PDU will be sent on a single engine cycle.

3.4.7.1 Output Channels

The CF application supports sending files to a configurable number of destinations. The output channels are configured through table parameters. Each channel has a pending queue, active queue and history queue. The pending queue reads may be enabled or disabled at anytime using the 'dequeue enable/disable' command. Each channel has a dedicated throttling semaphore, peer entity ID, message ID for outgoing PDUs and a configurable number of polling directories. File output transactions may occur simultaneously on different channels. The engine processes all active outgoing transactions in a round-robin fashion so as not to starve any one transaction. CF is not capable of prioritizing across channels.

3.4.7.2 Queuing Files For Output

There are three ways to request a file (or files) to be sent. Files can be queued by way of the file playback command, the directory playback command or through poll directory processing. The CF polling directory feature continually checks a directory for files and after detecting a new file in the directory, inserts a queue entry containing the file name (and other info) on the pending queue.

3.4.7.3 Priority

Each file-send transaction has an associated priority specified by the user. The priority of the transaction determines where it is inserted in the pending queue. High priority transactions get inserted toward the front of the queue. There are 256 levels of priority, zero being the highest. Priority is given as a command parameter for the playback file command and the playback directory command. For poll directory processing, each polling directory has an associated priority given as a table parameter. Please note that this priority applies only within a channel. CF does not support prioritization across channels. Prioritization across channels (if needed) would typically be implemented by the application receiving the PDUs. See section named Pending Queue Sorting Algorithm for more information.

3.4.7.4 Preserve Setting

When an outgoing file transaction is successfully complete, the user may want the file to be deleted by CF. The preserve setting allows the user to specify whether the file is deleted by CF or not. The preserve setting gives two choices, delete or keep. This setting is specified as a parameter in the playback file command, the playback directory command and on each polling directory. If a file transaction is not successful, the file will not be deleted.

3.4.7.5 Throttling Semaphore

Throttling outgoing PDUs may be necessary when the application that receives the outgoing PDUs (typically TO) needs to control the flow of packets. The throttling semaphore is a counting semaphore that is shared between another application and CF. Throttling may be configured as in-use or not-in-use on a per-channel basis. To configure as in-use, the receiving app must create a counting semaphore during initialization, using the name defined in the CF table. After creation, the receiving app must 'give' the semaphore each time it is ready to receive a PDU. On the CF side, CF attempts to get the semaphore ID by calling an OSAL function to Get-SemaphoreID-by-Name during CF initialization. The name defined in the table is given as a parameter to this call. CF has code to ensure that this call is executed after the receiving app initializes. If the attempt to Get-SemaphoreID-by-Name fails, then throttling on that channel is not-in-use and PDUs are sent whenever the engine has a PDU ready to output. If successful, each time the engine has a PDU to output, CF will attempt a non-blocking 'take' on the throttling semaphore. If the 'take' is successful, the green light counter in telemetry is incremented and the PDU is sent on the software bus. If the 'take' is not successful, the PDU is held by the engine, the red-light counter is incremented and the 'take' is called again on the next engine cycle.

The semaphore value is included in the CF housekeeping telemetry. This ensures that the semaphore does not lose or gain counts unexpectedly. If it were to gain a count, occasional pipe overflows would occur on TO's input pipe. If it were to lose a count, the system would not be fully utilizing the throughput. The value represents the number of empty buffers on the TO input pipe. To check that the semaphore value did not lose or gain counts, check the value only when there are no active transactions. The value should equal the depth of the TO input pipe. If the semaphore value becomes askew, it may be adjusted by the CF Semaphore Action command.

3.4.7.6 Polling Directories

When a polling directory is enabled, CF will periodically check the directory for files. If a file is found, a queue entry corresponding to the file is inserted on the pending queue. CF will place the entry on the queue only if the file is closed and not already pending or active on that channel. The polling rate is defined in the table and applies to all polling directories. If processor utilization is a concern, it is best to keep this rate as low as possible. Each channel has a configurable number of polling directories. Each polling directory has an enable, a class setting, a priority setting, a preserve setting, a peer entity ID and a destination directory.

Note: CF does not create these directories. They must be created before they can be enabled.
Subdirectories are not allowed in polling directories.

3.4.7.7 Auto Suspend Mode

The CF application contains an auto-suspend mode that may be useful when transaction timers need to be paused until a two-way communication link is present. For example, GPM uses this mode during the portion of the orbit when a downlink-only contact is present. Any transaction that is suspended will remain suspended until a 'resume' command is received. In the case of GPM, CF receives the resume-all command at the start of the two-way contact time.

The auto-suspend mode is disabled after a processor or power-on reset. It may be enabled or disabled at any time by command.

Suspending a transaction will pause timers for the transaction as well as prevent outgoing PDUs to be sent for the transaction. Incoming PDUs will be accepted and processed for suspended transactions.

When the EOF PDU is sent out, the CF application logs the transaction number in a suspend buffer. This buffer is deep enough to hold a configurable number (default value is 30) of transaction numbers. When the following wake-up command is received from the scheduler, CF will send a separate suspend command to the engine for each transaction number present in the buffer. There is a telemetry point (Low Free Mark) that indicates the utilization of the buffer. A value of 25 would indicate that only 5 of the 30 entries were used. This telemetry point should be monitored. If it falls below 5, it may require an adjustment to the compile-time configuration parameter, CF_AUTOSUSPEND_MAX_TRANS.

3.4.8 CF Memory Use

3.4.8.1 Memory Pool Heap

CF uses the CFE ES memory pool to manage a statically allocated heap of the size defined by the CF_MEMORY_POOL_BYTES configuration parameter. This heap is used to hold memory for queue entries. The life cycle of a queue entry begins when a request to queue a file for downlink is received. Or in the case of incoming transactions, the queue entry is allocated when the meta-data PDU is received by CF. With incoming transactions, the queue entry starts out on the incoming active queue, then the entry is moved to the history queue when the transaction completes. For outgoing transactions, the queue entry starts out on the pending queue, then moves to the active queue when the transaction begins, then moves to the history queue when the transaction is complete.

The history queue has a sliding window affect. When the queue is full and a new transaction needs to be added, the oldest transaction will be removed and returned to the heap, making room for the new transaction.

3.4.8.2 Incoming PDU Buffer

For incoming-file-transactions, CF statically allocates an incoming PDU buffer. The size of this buffer is defined by the platform configuration parameter, CF_INCOMING_PDU_BUF_SIZE. The incoming PDU's are copied from the Software Bus into this buffer and then passed to the engine.

3.4.8.3 Outgoing PDU Buffer

For outgoing-file-transactions, the engine statically allocates a buffer for PDUs. The size of this buffer is defined by platform configuration parameter, CF_OUTGOING_PDU_BUF_SIZE. The engine informs the CF app when it has a PDU ready to go out. In response to this, the CF app checks with the downlink app (e.g. TO) to see if it is ready to receive a PDU. This is done by the CF app trying to 'take' the throttling semaphore defined in the CF configuration table. If the CF app successfully 'takes' the semaphore, it gives a green light to the engine and the PDU is then released by the engine and sent to the software bus via the zero-copy delivery mode. There is a green light counter and a red light counter for each output channel in

telemetry. The green light counter shows the number of times TO has granted permission to CF to send a PDU. The red light counter shows the number of times TO has denied the sending of a PDU.

3.4.9 CF Efficiency

The CF application can be a processor intensive application. Some operating systems have significant overhead associated with file system operations. Opening and closing a file, querying the file system for file size, deleting the file, opening directories, looping through a directory list can consume a considerable amount of processor time. For this reason, transmitting small files at a high rate for long periods of time may be a worst-case timing scenario. File system overhead is less of an issue when file sizes are large. The terms 'large' and 'small' used here are relative to the downlink rate. With a downlink rate of 1 Mbps for example, a good file size would be 1 MByte or larger.

Also, it is best to keep the number of files on the pending queue to a minimum. When the number of files on the pending queue is high, (such as hundreds) prioritization and standard checking causes CF processing to be significant each time a file is added to the queue.

Polling directory processing is also subject to file system overhead. It is recommended that the rate of poll processing be kept low (table parameter not changeable by command) and unused polling directories should be disabled. During no-ground-contact time, polling directories should be disabled.

4 Operation of CF Software

4.1 Non-Adjustable CF Configuration Parameters

Although many CF configuration parameters are changeable by command, some are not. The parameters below are a subset of all non-changeable parameters. See the Appendices A.2, A.3 and A.4 for the full list of unchangeable parameters. Changing the following settings would require a parameter change plus a re-compile of either the CF application or the CF Configuration Table.

Pending Queue Depths	- 100 (Table Param)
History Queue Depths	- 20 (Table Param)
CF Pipe Depth	- 40 (Platform Config Param)
Max Simultaneous Transactions	- 100(Platform Config Param)
Polling Directory Checking	- Every 30 seconds (Table Param)
Max Transactions Suspended in 100ms	- 30 (Platform Config Param)

4.2 CF Telemetry Monitoring

CF Housekeeping Telemetry contains 3 resource variables that should be continuously monitored. Two variables show memory that is available and should therefore trigger an action if the value becomes less than a threshold. See CF Housekeeping Telemetry Packet defined in the Appendix for more detail.

1. LowFreeMark – Number of Auto Suspend Buffers available (low water mark). See section named Auto Suspend Mode for more detail.
2. LowMemoryMark – Number of bytes available for new queue entries (low water mark). See section named Memory Use.
3. Total Transactions In Progress – The max number of active transactions is defined by platform config parameter CF_MAX_SIMULTANEOUS_TRANSACTIONS.

4.3 CF Events

There are a number of nominal events that are sent by CF during file transfers. They are not filtered by default. The user has two choices for filtering these events. The commands below can be used for filtering during runtime or they may be filtered at CF startup by changing the cf_platform_cfg.h file.

```
; Outgoing Transaction Started
  /gc_evs_setbinfltrmask application="CF" event_id=103 filtermask=0xFFFF

  ; Outgoing Transaction Success
  /gc_evs_setbinfltrmask application="CF" event_id=21 filtermask=0xFFFF
```

```
; Transaction Suspended
/gc_evs_setbinfltrmask application="CF" event_id=26 filtermask=0xFFFF

; Transaction Resumed
/gc_evs_setbinfltrmask application="CF" event_id=27 filtermask=0xFFFF

; Freeze Command Received
/gc_evs_setbinfltrmask application="CF" event_id=46 filtermask=0xFFFF

; Thaw command Received
/gc_evs_setbinfltrmask application="CF" event_id=47 filtermask=0xFFFF

; Cancel/Abandon/Resume/Suspend Command Received
/gc_evs_setbinfltrmask application="CF" event_id=48 filtermask=0xFFFF
```

4.4 Ground Engine Commanding and Status

The commands sent to the CF application (and the status received from CF) apply only to the flight engine. The ground engine needs the same commanding and configuration parameter tuning that the flight engine receives. The user interface to the ground engine is done through STOL directives.

To view the ground engine configuration settings, enter the following at the STOL prompt:

```
page cfdp_config
```

To view the ground engine status enter the following at the STOL prompt:

```
page cfdp_status
```

To initiate a class 1, uplink transaction, the following directive can be executed at the stol prompt:

```
cfdp_dir  "put –class1 SrcPathAndFilename 0.24 DestPathAndFilename"
```

The '0.24' is the entity ID of the flight engine (CF).
For class 2 transfers, remove the '-class1' parameter.

To initiate a class 2, uplink transaction, the following directive can be executed at the stol prompt:

```
cfdp_dir  "put SrcPathAndFilename 0.24 DestPathAndFilename"
```

To adjust the ack_timeout of the ground engine, use the following directive:

```
cfdp_dir "ack_timeout=5"
```

NOTE: Be sure there is no space on either side of the equal sign.

Other CFDP ground engine command examples:

```
cfdp_dir  "put –class1 SrcPathAndFilename 0.24 DestPathAndFilename"
cfdp_dir  "put –class2 SrcPathAndFilename 0.24 DestPathAndFilename"
cfdp_dir "ack_timeout=5"
cfdp_dir "ack_limit=1"
cfdp_dir "nak_timeout=6"
cfdp_dir "nak_limit=1"
cfdp_dir "inactivity_timeout=60"
cfdp_dir "outgoing_file_chunk_size=200"
cfdp_dir "cancel 0.24_3"
cfdp_dir "cancel all"
cfdp_dir  "abandon 0.24_3"
cfdp_dir "suspend 0.24_3"
cfdp_dir "resume 0.24_3"
cfdp_dir "resume all"
cfdp_dir "freeze"
cfdp_dir "thaw"
cfdp_dir "?"
```

4.5 Operational Scenarios

This section describes how to control and adjust CF during common operational scenarios.

4.5.1 How to Stop File Transfers

At some point during operations, it may be necessary to stop the flow of file data as quickly as possible. Some of the steps mentioned below may not be needed. The first step may be to disable dequeuing on all pending queues. This will ensure that no new files begin transferring during the process. Then disable polling on all channels and purge the pending queues. At this point any new files will be kept safely in the file system directories. Next, abort the files in progress by abandoning all files. This can be done with a single command to the flight – abandon all. Abandoning the transactions will not have an adverse affect on the files, but the file transfers will not pick-up where they left off after the flow of file data is resumed. The files will be closed and left in the original directory on board. The flight-side abandon just mentioned will leave the ground engine waiting for PDUs that will never be received. It will be necessary to do a ground-side abandon-all as well, otherwise ground timeouts will likely occur. At this point there will likely be PDUs in the TO downlink pipes that need to be flushed. It would be best to let TO operate normally until its input pipes are flushed. See the section below named "Flushing the TO Input Pipes".

4.5.2 Flushing the TO Input Pipes

There are times during operations when it may be necessary to flush the file PDU's that are present on the TO input pipes. This would be the case if the flight software was entering safehold or when the spacecraft would temporarily lose contact while slewing from one communication link to another. In the case of the FSW entering safehold, flushing the pipes may be needed so that high priority file data such as a housekeeping file does not get blocked or delayed by file-data left over from an abandon transaction. In the case of slewing, it may be necessary to flush the pipes before slewing so that no file data is lost during the transition.

To flush the pipes allow enough time for the operation to be done naturally. For instance, in the case of entering safehold it may be necessary to delay the downlink-rate change until the pipes are flushed. For the case of slewing, it may be necessary to delay the slew after the freeze-flight-engine command is executed. The pipes are flushed when the CF semaphore value (in housekeeping telemetry) shows the TO pipe depth. This telemetry point shows the number of empty buffers on the pipe.

The time needed to flush the pipes is dependent on the commanded downlink rate and the amount of real-time traffic (which has a higher priority than file data). For example, there are three pipes that hold PDUs at the input of TO. Each pipe is capable of holding 30 PDUs. Assume each VCDU is 1800 bytes and holds one PDU. As a worst-case timing number, consider that all three pipes are full. This corresponds to 162000 bytes of data that must be flushed. If the downlink rate was 230Kbps and the real-time packets used 30% of that (leaving 70% for file data), then the time to flush the pipes would be about 8 seconds. 230Kbps = 28750 total bytes per second. 28750 * 70% = 20125 file bytes per second. 162000 / 20125 = 8.04 seconds. It would be safe to add a few seconds due to unforeseen overhead related to the downlink.

Appendix A - CF Configuration Parameters

This section shows the default parameters that are delivered with a release of the CF application. The parameters listed in this section are meant to be adjusted by the project.

A.1 CF Message IDs

```
/**************************
** CF Command Message IDs
**************************/
#define CF_CMD_MID                  0x18B3
#define CF_SEND_HK_MID              0x18B4
#define CF_WAKE_UP_REQ_CMD_MID      0x18B5
#define CF_SPARE1_CMD_MID           0x18B6
#define CF_INCOMING_PDU_MID         0x1FFD


/**************************
** CF Telemetry Message IDs
**************************/
#define CF_HK_TLM_MID               0x08B0
#define CF_TRANS_TLM_MID            0x08B1
#define CF_SPARE_TLM_MID            0x08B2
#define CF_CONFIG_TLM_MID           0x08B3

/*
** NOTE: the definitions below are NOT used by the code directly. The code uses
** the MsgId defined in the CF table. The CF table should use these macro
** definitions.
*/
#define CF_SPACE_TO_GND_PDU_MID     0x0FFD
```

A.2 CF Platform Configuration Parameters

CF uses the following compile-time parameters. The values shown cannot be changed by command. Changing platform configuration parameters involves rebuilding the application.

This section shows the CF platform configuration values that are delivered as defaults with the application.

```
/**
**  \cfcfg Application Pipe Depth
**
**  \par Description:
**       Dictates the pipe depth of the cf command pipe.
**
**  \par Limits:
**            The minimum size of this parameter is 1
**       The maximum size dictated by cFE platform configuration
**            parameter is CFE_SB_MAX_PIPE_DEPTH
*/
#define CF_PIPE_DEPTH                    40

/**
**  \cfcfg Application Pipe Name
**
**  \par Description:
**       Dictates the pipe name of the cf command pipe.
**
**  \par Limits:
**
*/
#define CF_PIPE_NAME                     "CF_CMD_PIPE"

/**
**  \cfcfg Maximum Simultaneous Transactions
**
**  \par Description:
**      Dictates max number of transactions (uplink and downlink)
**      that can be in progress at any given time.
**
**  \par Limits:
**
*/
#define CF_MAX_SIMULTANEOUS_TRANSACTIONS    100

/**
**  \cfcfg Uplink PDU Data Buffer Size
```

```
**
**  \par Description:
**       This parameter sets the statically allocated size (in bytes) of the
**       incoming PDU buffer. This buffer will be used to hold the pdu hdr and
**       data portion of the incoming PDUs. Incoming PDUs are enclosed in a CCSDS
**       packet. This parameter should not include the size of the CCSDS pkt hdr.
**
**  \par Limits:
**       Must be greater than or equal to the sum of the ground engine parameter
**       outgoing-file-chunk-size, pdu hdr size and the 4 bytes of 'offset' in
**       file-data pdus. Upper limit of 64K derived from 16 bit PDU header field
**       named 'PDU Data Field Length'.
**       This parameter must be less-than or equal-to the outgoing pdu buffer.
**
**
*/
#define CF_INCOMING_PDU_BUF_SIZE            512


/**
**  \cfcfg Outgoing PDU Data Buffer Size
**
**  \par Description:
**       This parameter sets the statically allocated size (in bytes) of the
**       outgoing PDU buffer. This buffer will be used to hold the pdu hdr and
**       data portion of the outgoing PDUs. Outgoing PDUs are enclosed in a CCSDS
**       packet. This parameter should not include the size of the CCSDS pkt hdr.
**
**  \par Limits:
**       This parameter will put an upper limit on the table parameter
**       'outgoing_file_chunk_size'. The max 'outgoing_file_chunk_size' allowed
**       will be CF_OUTGOING_PDU_BUF_SIZE - (12 + 4) The 12 and 4 are pdu hdr
**       size and offset field in file-data pdu, respectively.
**       This parameter has an upper limit of 64K derived from 16 bit PDU header
**       field named 'PDU Data Field Length'.
**       This parameter must be greater-than or equal-to the incoming pdu buffer.
**
**
*/
#define CF_OUTGOING_PDU_BUF_SIZE            2048


/**
**  \cfcfg Path name and file prefix of the engine temp files
**
**  \par Description:
**       The receiving engine constructs all files in a temporary file. This
**       parameter specifies the path and base filename of the temporary files.
**       The engine appends a sequence number to this parameter to get a complete
**       filename.
**
```

```
**  \par Limits:
**       - The length of this string, including the NULL terminator cannot exceed
**           the #OS_MAX_PATH_LEN value.
**       - The last character should not be a slash.
**
*/
#define CF_ENGINE_TEMP_FILE_PREFIX          "/ram/cftmp"

/**
**  \cfcfg Name of the CF Configuration Table
**
**  \par Description:
**       This parameter defines the name of the CF Configuration Table.
**
**  \par Limits
**       The length of this string, including the NULL terminator cannot exceed
**       the #OS_MAX_PATH_LEN value.
*/
#define CF_CONFIG_TABLE_NAME                "ConfigTable"


/**
**  \cfcfg CF Configuration Table Filename
**
**  \par Description:
**       The value of this constant defines the filename of the CF Config Table
**
**  \par Limits
**       The length of this string, including the NULL terminator cannot exceed
**       the #OS_MAX_PATH_LEN value.
*/
#define CF_CONFIG_TABLE_FILENAME            "/cf/cf_cfgtable.tbl"


/**
**  \cfcfg Number of Input Channels
**
**  \par Description:
**       Defines the number of input channels
**       defined in the configuration table. Input channels were added to the
**       design to support class 2 file receives from multiple peers. It is
**       necessary for the code to know what output channel should be used for
**       responses (ACK-EOF,NAK, etc) of incoming, class 2 transactions.
**       Each input channel has a dedicated MsgId for incoming PDUs and an output
**       channel for responses of class 2, file-receive transactions.
**
**  \par Limits
**       Lower Limit of 1, Upper limit of 255.
**
*/
#define CF_NUM_INPUT_CHANNELS               1
```

```
/**
**  \cfcfg Max Number of Playback Output Channels
**
**  \par Description:
**       Defines the max number of playback output channels that may ever be
**       defined in the configuration table. Refer to the configuration table for
**       more details about playback output channels.
**
**  \par Limits
**       Lower Limit of 1, Upper limit of 255.
**
**  \par Notes:
**       The CF configuration table must have an entry for this number of
**       playback channels, but some may be marked as not-in-use. This saves
**       having to recompile and reload a new CF Application when a playback
**       channel is added.
*/
#define CF_MAX_PLAYBACK_CHANNELS            2


/**
**  \cfcfg Max Number of Polling Directories per Playback Output Channel
**
**  \par Description:
**       Defines the max number of polling directories that may ever be defined
**       in the configuration table. A polling directory is a directory that
**       is periodically checked for playback files. Files found in the polling
**       directory are immediately placed on the playback pending queue for
**       downlink.
**
**  \par Limits:
**       Lower limit of 1, Upper limit of 255.
**
**  \par Notes:
**       The CF configuration table must have an entry for this number of polling
**       directories, but some may be marked as not-in-use. This saves having to
**       recompile and reload a new CF Application when a polling directory is
**       added.
**
*/
#define CF_MAX_POLLING_DIRS_PER_CHAN        8

/**
**  \cfcfg Number of bytes in the CF Memory Pool
**
**  \par Description:
**       The CF memory pool contains the memory needed to hold information for
**       each transaction. The info for each transaction is defined by a
**       CF_QueueEntry_t. The number of CF_QueueEntry_t's needed is based on:
**
**       UplinkHistoryQDepth + CF_MAX_SIMULTANEOUS_TRANSACTIONS +
```

```
**     ((CF_MAX_PLAYBACK_CHANNELS * (PendingQDepth + HistoryQDepth))
**
**     Lower case variables are defined in config table, upper case params are
**     defined in platform config file (cf_platform_cfg.h)
**
**     See CF Housekeeping page for memory utilization details
**
**
**  \par Limits
**      Lower Limit of 256, Upper limit of 4 Gigabytes
*/
#define CF_MEMORY_POOL_BYTES              32768


/**
**  \cfcfg Default Queue Information Filename
**
**  \par Description:
**      The value of this constant defines the filename used to store the CF
**      queue information.  This filename is used only when no filename is
**      specified in the command.
**
**  \par Limits
**      The length of each string, including the NULL terminator cannot exceed
**      the OS_MAX_PATH_LEN value.
*/
#define CF_DEFAULT_QUEUE_INFO_FILENAME      "/ram/cf_queue_info.dat"

/**
**  \cfcfg CF Event Filtering
**
**  \par Description:
**      This group of configuration parameters dictates what CF events will be
**      filtered through EVS. The filtering will begin after the CF app
**      initializes and stay in effect until changed via EVS command.
**      Mark all unused event Id values and mask values to zero
**      eg.    #define CF_FILTERED_EVENT1         0
**             #define CF_FILTER_MASK1            0
**      To filter the event, set the mask value to CFE_EVS_FIRST_ONE_STOP
**      To disable filtering of the event, set mask value to CFE_EVS_NO_FILTER
**
**  \par Limits
**      These parameters have a lower limit of 0 and an upper limit of 65535.
*/

#define CF_FILTERED_EVENT1              CF_IN_TRANS_START_EID
#define CF_FILTER_MASK1                 CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT2              CF_IN_TRANS_OK_EID
#define CF_FILTER_MASK2                 CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT3              CF_OUT_TRANS_START_EID
```

```
#define CF_FILTER_MASK3             CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT4            CF_OUT_TRANS_OK_EID
#define CF_FILTER_MASK4             CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT5            0
#define CF_FILTER_MASK5             CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT6            0
#define CF_FILTER_MASK6             CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT7            0
#define CF_FILTER_MASK7             CFE_EVS_NO_FILTER

#define CF_FILTERED_EVENT8            0
#define CF_FILTER_MASK8             CFE_EVS_NO_FILTER


/**
**  \cfcfg Time to wait for all apps to be started (in milliseconds)
**
**  \par Description:
**       Dictates the timeout for the #CFE_ES_WaitForStartupSync call that
**       CF uses to ensure that TO or the downlink App has completed it's
**       initialization which includes creating the semaphore needed by CF.
**
**  \par Limits
**       This parameter can't be larger than an unsigned 32 bit
**       integer (4294967295).
**
**       This should be greater than or equal to the Startup Sync timeout for
**       any application in the Application Monitor Table.
*/
#define CF_STARTUP_SYNC_TIMEOUT   5000


/**
**  \cfcfg Use fixed size packets (for outgoing PDUs) or not.
**
**  \par Description:
**       When sending PDUs, CF can be configured to place the PDUs in fixed-size
**       pkts or let the PDU size determine the pkt size. The value defined
**       must correspond to the CCSDS Total Message size which includes the PDU
**       header, the CCSDS header and data.
**       Set this value to 0 for variable pkt sizes.
**
**  \par Limits
**       This parameter can't be larger than CFE_SB_MAX_SB_MSG_SIZE (typically
**       set to 32K or 64K bytes)
**
**       If non-zero, this should be greater than or equal to the size needed to
**       hold the largest PDU expected to be sent by the engine (typically a
```

```
**      file data PDU which is derived from the CF table cfg param
**      "OutgoingFileChunkSize"). This value must also include the CCSDS header
**      size.
*/
#define CF_SEND_FIXED_SIZE_PKTS   0


/**
**  \cfcfg Auto-Suspend, max transactions to suspend
**
**  \par Description:
**      When auto suspend is enabled, after EOF is sent the transaction number
**      is logged in a buffer. The buffer size is defined in this parameter.
**      After the following wakeup cmd is received, CF will check this buffer
**      for transactions to suspend. They cannot be suspended at the time the
**      EOF is sent because the engine is not designed to be re-entrant.
**
**  \par Limits
**      This parameter must be greater than zero and can't be larger than an
**      unsigned 32 bit integer (4294967295).
*/
#define CF_AUTOSUSPEND_MAX_TRANS  30



/** \cfcfg Mission specific version number for CF application
**
**  \par Description:
**      An 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 "cf_version.h".
**
**  \par Limits:
**      Must be defined as a numeric value that is greater than
**      or equal to zero.
*/
#define CF_MISSION_REV      0


/** \cfcfg Compile-time debug switch for CF application
**
**  \par Description:
**     CF_DEBUG should NOT be defined under normal conditions. It is to be used
**     as a safety net during development, when a uart terminal is connected to
**     the processor. When the code is compiled with CF_DEBUG defined, the code
**     will issue OS_printfs in areas that would otherwise be quiet. It would also be
**     possible to view the queue data, table data and configuration data from the shell
**     via CF_ShowQs, CF_ShowTbl and CF_ShowCfg
**
```

```
**  \par Limits:
**       Must be defined or commented out.
*/
/* #define CF_DEBUG */
```

A.3 CF Mission Configuration Parameters

The mission configuration parameters are performance markers used for timing measurements.
The values shown cannot be changed by command. Changing mission configuration parameters involves rebuilding the application.

```
#define CF_APPMAIN_PERF_ID      42
#define CF_FILESIZE_PERF_ID     41
#define CF_FOPEN_PERF_ID        13
#define CF_FCLOSE_PERF_ID       14
#define CF_FREAD_PERF_ID        15
#define CF_FWRITE_PERF_ID       16
#define CF_REDLIGHT_PERF_ID     17
#define CF_CYCLE_ENG_PERF_ID 18
#define CF_QDIRFILES_PERF_ID    19
```

A.4 CF Configuration Table Parameters

The CF Configuration table shown here is the default table delivered with CF. The project will need to customize this table to meet the needs of the mission. This table contains two types of parameters, changeable-by-command and static (denoted by the !!!). CF validates and loads the configuration table during initialization only. CF does not check for table loads during runtime. Any attempt to load this table during runtime must be aborted. If the load attempt is not aborted, commands to dump or validate the table will wait forever. Any table parameter that becomes changed by command will cause the table checksum to change. Reserved and unused entries are part of the actual table but are not shown below.

Static parameters that are not changeable-by-command are denoted with a triple bang (!!!).

```
cf_config_table_t     CF_ConfigTable =
{

   "CF Default Table",!!!/* TableIdString */
   2, !!!     /* TableVersion (integer)   */
   4, !!!     /* NumEngCyclesPerWakeup    */
   2, !!!     /* NumWakeupsPerQueueChk, applies to all channels    */
   4, !!!     /* NumWakeupsPerPollDirChk, apples to all polling directories  */
   100, !!! /* UplinkHistoryQDepth      */
   "10",     /* AckTimeout (seconds, entered as string) */
   "2",      /* AckLimit (number of timeouts needed for failure, string)      */
   "5",      /* NakTimeout (seconds, string)           */
   "3",      /* NakLimit ((number of timeouts needed for failure, string)      */
   "20",     /* InactivityTimeout (seconds, string)     */
   "200",    /* OutgoingFileChunkSize (bytes, string) */
   "no",     /* SaveIncompleteFiles (yes,no, string) */
   "0.24",  /* Flight EntityId - 2 byte dotted-decimal string eg. "0.255"*/

   { /* Input Channel Array */
      { /* Input Channel 0 */

         CF_INCOMING_PDU_MID, !!!
         0, !!!      /* Output Chan for Class 2 Uplink Responses, ACK-EOF,Nak,Fin etc) */
         0,          /* spare */

      }, /* end Input Channel 0 */
   }, /* end Input Channel Array */

   {   /* Playback Channel Array */
      {   /* Playback Channel #0 */
         CF_ENTRY_IN_USE, !!!                  /* Playback Channel Entry In Use */
         CF_ENABLED,                           /* Dequeue Enable */
         CF_SPACE_TO_GND_PDU_MID, !!!  /* Space To Gnd PDU MsgId */
         100, !!!                              /* Pending Queue Depth */
         100, !!!                              /* History Queue Depth */
         "TOPBOutputChan0",   !!!              /* Playback Channel Name   */
         "CFTOSemId",  !!!                     /* Handshake Semaphore Name   */
```

```
{   /* Polling Directory Array */

  { /* Chan 0 Polling Directory 0 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    5,                        /* Priority */
    CF_KEEP_FILE,             /* Preserve files after successful transfer? */
    "0.23",                   /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch0poll0/",          /* SrcPath, no spaces, fwd slash at end */
    "cftesting/",             /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 0 */

  { /* Chan 0 Polling Directory 1 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
    CF_ KEEP _FILE,           /* Preserve files after successful transfer? */
    "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/ cf/ch0poll1/",         /* SrcPath, no spaces, fwd slash at end */
    "/gnd/",                  /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 1 */

  { /* Chan 0 Polling Directory 2 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    5,                        /* Priority */
    CF_DELETE_FILE,           /* Preserve files after successful transfer? */
    "0.23",                   /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch0poll2/",          /* SrcPath, no spaces, fwd slash at end */
    " cftesting/",            /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 2 */

  { /* Chan 0 Polling Directory 3 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
    CF_KEEP_FILE,             /* Preserve files after successful transfer? */
    "0.23",                   /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch0poll3/",          /* SrcPath, no spaces, fwd slash at end */
    "/gnd/",                  /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 3 */

  { /* Chan 0 Polling Directory 4 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    5,                        /* Priority */
```

```
            CF_KEEP_FILE,              /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch0poll4/",           /* SrcPath, no spaces, fwd slash at end */
            "cftesting/",              /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 4 */

          { /* Chan 0 Polling Directory 5 */
            CF_ENTRY_IN_USE, !!!       /* Poll Directory In Use or Not */
            CF_DISABLED,               /* Enable State */
            1,                         /* Class (1 or 2)*/
            0,                         /* Priority */
            CF_ KEEP _FILE,            /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch0poll5/",           /* SrcPath, no spaces, fwd slash at end */
            "/gnd/",                   /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 5 */

          { /* Chan 0 Polling Directory 6 */
            CF_ENTRY_IN_USE, !!!       /* Poll Directory In Use or Not */
            CF_DISABLED,               /* Enable State */
            1,                         /* Class (1 or 2)*/
            5,                         /* Priority */
            CF_DELETE_FILE,            /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch0poll6/",           /* SrcPath, no spaces, fwd slash at end */
            " cftesting/",             /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 6 */

          { /* Chan 0 Polling Directory 7 */
            CF_ENTRY_IN_USE, !!!       /* Poll Directory In Use or Not */
            CF_DISABLED,               /* Enable State */
            1,                         /* Class (1 or 2)*/
            0,                         /* Priority */
            CF_KEEP_FILE,              /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch0poll7/",           /* SrcPath, no spaces, fwd slash at end */
            "/gnd/",                   /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 7 */

        }, /* End Polling Directory Array */

      },  /* End Playback Channel #0 */


{   /* Playback Channel #1 */
        CF_ENTRY_IN_USE, !!!                /* Playback Channel Entry In Use */
        CF_DISABLED,                        /* Dequeue enable for pending queue*/
        CF_SPACE_TO_GND_PDU_MID, !!!  /* Space To Gnd PDU MsgId */
        100, !!!                            /* Pending Queue Depth */
        100, !!!                            /* History Queue Depth */
        "TOPBOutputChan1",   !!!            /* Playback Channel Name   */
        "CFTOSemId",  !!!                   /* Handshake Semaphore Name   */
```

```
{   /* Polling Directory Array */

  { /* Chan 1 Polling Directory 0 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
    CF_DELETE_FILE,           /* Preserve files after successful transfer? */
    "0.23",                   /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch1poll0/",          /* SrcPath, no spaces, fwd slash at end */
    "cftesting/",             /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 0 */

  { /* Chan 1 Polling Directory 1 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
    CF_ DELETE _FILE,         /* Preserve files after successful transfer? */
   "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch1poll1/",          /* SrcPath, no spaces, fwd slash at end */
    "cftesting/",             /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 1 */

  { /* Chan 1 Polling Directory 2 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
    CF_DELETE_FILE,           /* Preserve files after successful transfer? */
    "0.23",                   /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch1poll2/",          /* SrcPath, no spaces, fwd slash at end */
    "cftesting/",             /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 2 */

  { /* Chan 1 Polling Directory 3 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
    CF_ DELETE _FILE,         /* Preserve files after successful transfer? */
    "0.23",                   /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
    "/cf/ch1poll3/",          /* SrcPath, no spaces, fwd slash at end */
    "cftesting/",             /* DstPath, no spaces, fwd slash at end */
  },/* End Polling Directory 3 */

  { /* Chan 1 Polling Directory 4 */
    CF_ENTRY_IN_USE, !!!      /* Poll Directory In Use or Not */
    CF_DISABLED,              /* Enable State */
    1,                        /* Class (1 or 2)*/
    0,                        /* Priority */
```

```
            CF_ DELETE _FILE,          /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch1poll4/",           /* SrcPath, no spaces, fwd slash at end */
            "cftesting/",              /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 4 */

          { /* Chan 1 Polling Directory 5 */
            CF_ENTRY_IN_USE, !!!       /* Poll Directory In Use or Not */
            CF_DISABLED,               /* Enable State */
            1,                         /* Class (1 or 2)*/
            0,                         /* Priority */
            CF_ DELETE _FILE,          /* Preserve files after successful transfer? */
            "0.23",                     /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch1poll5/",           /* SrcPath, no spaces, fwd slash at end */
            "cftesting/",              /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 5 */

          { /* Chan 1 Polling Directory 6 */
            CF_ENTRY_IN_USE, !!!       /* Poll Directory In Use or Not */
            CF_DISABLED,               /* Enable State */
            1,                         /* Class (1 or 2)*/
            0,                         /* Priority */
            CF_DELETE_FILE,            /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch1poll6/",           /* SrcPath, no spaces, fwd slash at end */
            "cftesting/",              /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 6 */

          { /* Chan 1 Polling Directory 7 */
            CF_ENTRY_IN_USE, !!!       /* Poll Directory In Use or Not */
            CF_DISABLED,               /* Enable State */
            1,                         /* Class (1 or 2)*/
            0,                         /* Priority */
            CF_ DELETE _FILE,          /* Preserve files after successful transfer? */
            "0.23",                    /* Gnd EntityId - 2 byte dotted-decimal string eg. "0.255"*/
            "/cf/ch1poll7/",           /* SrcPath, no spaces, fwd slash at end */
            "cftesting/",              /* DstPath, no spaces, fwd slash at end */
          },/* End Polling Directory 7 */

        }, /* End Polling Directory Array */
      },  /* End Playback Channel #1 */
    },  /* End Playback Channel Array */
}; /* End CF_ConfigTable */
```

Appendix B - Commands

B.1 NOOP (No Operation)

Command Name:	NOOP (No Operation)
Mnemonic:	/SCX_CPU1_CF_NOOP
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0
Data Field Length:	0
CCSDS Format:	18B3 C000 0001 0000
Data Fields: None.	
Operation:	This command verifies that the CF Application responds to commands; it sends an informational event containing the application version number.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments.

B.2 RESET COUNTERS

Command Name:	RESET COUNTERS
Mnemonic:	/SCX_CPU1_CF_RESETCTRS
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	1
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_RESETCTRS All (must enter union name in lieu of value=0)
CCSDS Format:	18B3 C000 0005 0100 aabb bbbb
Data Fields: Union aa - All (0), Command (1), Fault (2), Incoming (3), Outgoing (4) bb – spare byte	
Operation:	This command resets the specified counters. Union named 'Incoming' refers to the counters related to uplink transactions, 'Outgoing' is downlink transactions.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter becomes zero. [SCX_CPU1_CF_CMDEC] Command error counter becomes zero.

B.3 PLAYBACK FILE

Command Name:	PLAYBACK FILE
Mnemonic:	/SCX_CPU1_CF_PLAYBACKFILE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	2
Data Field Length:	148
Syntax Example:	/SCX_CPU1_CF_PLAYBACKFILE CLASS_2 Chan_1 PRIORITY=9 DELETE_FILE PEERENTITYID="0.23" SRCFILENAME="/sdr/file.sci" DESTFILENAME= "/groundpath/file.sci"
CCSDS Format:	18B3 C000 0095 0200 aabb ccdd eeee eeee eeee eeee eeee eeee eeee ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg

Data Field: Union aa – Class_1(1), Class_2(2)
Data Field: Union bb – Chan_0(0), Chan_1(1)
Data Field: cc – Priority – 00 (highest priority) – 0xff (lowest priority)
Data Field: Union dd – Delete_File(0), Keep_File(1) – Action taken after a successful transfer
Data Field: ee – PeerEntityID - 16 byte field – string – Entity ID of ground engine to receive file
Data Field: ff – SrcFilename - 64 byte field – string – Absolute path with name of file to be sent
Data Field: gg – DestFilename - 64 byte field – string – Absolute path or path relative to gnd engine default path. Example "cffiles/" string will have file stored in $WORK/image/cffiles directory.

Operation:	This command is used to queue the specified file.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments. Pending queue file count will increment.

B.4 PLAYBACK DIRECTORY

Command Name:	PLAYBACK DIRECTORY
Mnemonic:	/SCX_CPU1_CF_PLAYBACKDIR
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	3
Data Field Length:	148
Syntax Example:	/SCX_CPU1_CF_PLAYBACKFILE CLASS_2 Chan_1 PRIORITY=9 DELETE_FILE PEERENTITYID="0.23" SRCPATH="/sdr/gmi/hk/" DSTPATH= "cffiles/"
CCSDS Format:	18B3 C000 0095 0300 aabb ccdd eeee eeee eeee eeee eeee eeee eeee eeee ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff ffff gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg gggg
Data Field: Union aa – Class_1(1), Class_2(2)	
Data Field: Union bb – Chan_0(0), Chan_1(1)	
Data Field: cc – Priority – 00 (highest priority) – 0xff (lowest priority)	
Data Field: Union dd – Delete_File(0), Keep_File(1) – Action taken after a successful transfer	
Data Field: ee – PeerEntityID - 16 byte field – string – Entity ID of ground engine to receive file	
Data Field: ff – SrcPath - 64 byte field – string – absolute path of directory to playback. String must end with forward slash '/' character.	
Data Field: gg – DstPath - 64 byte field – string – absolute path or path relative to gnd engine default path. Example "cffiles/" string will have file stored in $WORK/image/cffiles directory. String must end with forward slash '/' character.	
Operation:	This command is used to queue all files in the specified (SRCPATH) directory.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments. Pending queue will receive one entry for each file queued.

B.5 FREEZE

Command Name:	FREEZE
Mnemonic:	/SCX_CPU1_CF_FREEZE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	4
Data Field Length:	0
CCSDS Format:	18B3 C000 0001 0400
Data Fields: none	
Operation:	This command will freeze the flight engine. The flight engine will pause timers for all active transactions and stop outputting pdus. The freeze command is typically applied to both the flight and ground peers at nearly the same time.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter becomes zero. [SCX_CPU1_CF_ENGINEFLAGS] Bit 0, zero=thawed, one=frozen CF_FREEZE_CMD_EID Informational event is sent.

B.6 THAW

Command Name:	THAW
Mnemonic:	/SCX_CPU1_CF_THAW
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	5
Data Field Length:	0
CCSDS Format:	18B3 C000 0001 0500
Data Fields: none	
Operation:	This command is used to thaw all active transactions that were frozen on the flight engine (See Freeze Command). The thaw command is typically applied to both the flight and ground peers at nearly the same time. NOTE: A suspended transaction does not resume when a thaw command is received. Suspended transactions must be resumed. See Suspend and Resume commands.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter becomes zero. [SCX_CPU1_CF_ENGINEFLAGS] Bit 0, zero=thawed, one=frozen CF_THAW_CMD_EID Informational event is sent.

B.7 SUSPEND

Command Name:	SUSPEND
Mnemonic:	/SCX_CPU1_CF_SUSPEND
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	6
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_SUSPEND TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 0600 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

Data Fields: aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The transaction must be active. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. Entering the Transaction ID is less ambiguous. The string – "All" is allowed and will suspend all active transactions. The lettering of the word 'All' is not case sensitive.

Operation:	This command is used to suspend one or all transactions on the flight side. The flight engine will pause timers for the transaction(s) and stop outputting pdus related to the transaction(s). The suspend command is typically applied to both the flight and ground peers at nearly the same time. NOTE 1: When a suspended transaction is cancelled, the cancel does not take affect until the transaction is resumed. NOTE 2: Suspending an outgoing transaction before EOF is sent, will pause the flow of PDUs on that channel. This happens because the next file is started when the current file EOF is sent. See the CF_KICKSTART_CC command description for more detail. If a user wishes to stop the current transaction (before the EOF is sent) and still allow the next pending file to begin, the current transaction should be cancelled (or abandoned) in lieu of being suspended. Canceling is always a better option than abandoning.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. CF_SUSPEND_CMD_EID Informational event is sent. Send SCX_CPU1_CF_QuickStatus command to verify that the transaction is suspended.

B.8 RESUME

Command Name:	RESUME
Mnemonic:	/SCX_CPU1_CF_RESUME
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	7
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_RESUME TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 0700 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

Data Fields: aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The transaction must be active. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. Entering the Transaction ID is less ambiguous. The string – "All" is allowed and will resume all active transactions. The lettering of the word 'All' is not case sensitive.

Operation:	This command is used to resume a suspended transaction or all transactions on the flight side.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. CF_RESUME_CMD_EID Informational event is sent. Number of Suspended transactions decrements. Send SCX_CPU1_CF_QuickStatus command to verify that the transaction is no longer suspended.

B.9 CANCEL

Command Name:	CANCEL
Mnemonic:	/SCX_CPU1_CF_CANCEL
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	8
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_CANCEL TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 0800 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

Data Fields aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The transaction must be active. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. Entering the Transaction ID is less ambiguous. The string – "All" is allowed and will cancel all active transactions. The lettering of the word 'All' is not case sensitive.

Operation:	This command is used to cancel a transaction or all transactions on the flight side. The cancel command should be sent to the source entity only. For example, uplink transactions should be cancelled at the ground engine. The CF application should not receive a cancel command in this case. The CF application will learn of the cancel request through the protocol messages. Downlink transactions and outgoing transactions (with respect to CF) should be cancelled by sending this CF cancel command. NOTE: If a Cancel command is received by CF on an outgoing transaction that is suspended, the cancel does not take affect until the transaction is resumed.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. [SCX_CPU1_CF_CancelNum] Total cancelled transactions increments. [CF_TotalFailedTrans] Total failed transactions increments. CF_CANCEL_CMD_EID Informational event is sent. Send SCX_CPU1_CF_QuickStatus command to verify that the transaction has been cancelled.

B.10 ABANDON

Command Name:	ABANDON
Mnemonic:	/SCX_CPU1_CF_ABANDON
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	9
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_ABANDON TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 0900 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

Data Fields: Data Fields: aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The transaction must be active. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. Entering the Transaction ID is less ambiguous. The string – "All" is allowed and will abandon all active transactions. The lettering of the word 'All' is not case sensitive.

Operation:	This command is used to abandon a single transaction or all transactions on the flight side. The abandon command is typically applied to both the flight and ground peers at nearly the same time. After receiving the abandon command, the engine terminates the transaction and no longer attempts to communicate with the peer. NOTE: Unlike the cancel command, if a suspended transaction is abandoned, the transaction will be abandoned at the time the abandon command is received. Likewise, if a frozen transaction is abandoned, the transaction will be abandoned when the abandoned cmd is received.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. [SCX_CPU1_CF_TotalAbandTrans] Total abandoned transactions increments. [CF_TotalFailedTrans] Total failed transactions increments. CF_ABANDON_CMD_EID Informational event is sent. Send SCX_CPU1_CF_QuickStatus command to verify that the transaction has been abandoned.

B.11 SET MIB PARAMETER

Command Name:	SET MIB PARAMETER
Mnemonic:	/SCX_CPU1_CF_SETMIBPARAM
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x0A, 10 decimal
Data Field Length:	48
CCSDS Format:	18B3 C000 0031 0A00 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb
Data Fields: aa – Param - 32 byte string – Engine MIB parameter such as ACK_TIMEOUT, ACK_LIMIT, NAK_TIMEOUT, NAK_LIMIT, INACTIVITY_TIMEOUT, OUTGOING_FILE_CHUNK_SIZE, SAVE_INCOMPLETE_FILES	
Data Field: bb – Value - 16 byte string – timeout values are number of seconds, limit values are number of timeouts, outgoing_file_chunk_size is number of bytes and save_incomplete_files is yes or no (not case sensitive).	
Operation:	This command is used to change the flight engine Message Information Base (MIB). The MIB is a term used in the CCSDS blue book that can be interpreted as the engine configuration parameters. The command has two command parameters, Param indicates which parameter to change, and Value indicates the new setting. The Asist database contains a command for each parameter to set. This eliminates the need to know the exact syntax of the parameter name required by the engine.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments. Informational event CF_SET_MIB_CMD_EID will be sent in response to command. Send SCX_CPU1_CF_GETMIBPARAM to view the new setting in an event. CF Configuration table will contain the new setting, dump table to verify. Send SCX_CPU1_CF_SENDCFG command to receive a packet with the new settings.

B.12 GET MIB PARAMETER

Command Name:	GET MIB PARAMETER
Mnemonic:	/SCX_CPU1_CF_GETMIBPARAM
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x0B, 11 decimal
Data Field Length:	32
CCSDS Format:	18B3 C000 0021 0B00 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
Data Fields: aa – Param - 32 byte string – Engine MIB parameter such as ACK_TIMEOUT, ACK_LIMIT, NAK_TIMEOUT, NAK_LIMIT, INACTIVITY_TIMEOUT, OUTGOING_FILE_CHUNK_SIZE, SAVE_INCOMPLETE_FILES	
Operation:	This command is used to verify a parameter value of the flight engine Message Information Base (MIB). The MIB is a term used in the CCSDS blue book that can be interpreted as the engine configuration parameters. The command has one parameter, Param indicates which parameter value to display in the event. The Asist database contains a command for each parameter to get. This eliminates the need to know the exact syntax of the parameter name required by the engine.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments. Informational event CF_GET_MIB_CMD_EID will be sent in response to command.

B.13 SEND TRANSACTION DIAGNOSTIC PACKET

Command Name:	SEND TRANSACTION DIAGNOSTIC PACKET
Mnemonic:	/SCX_CPU1_CF_SENDTRANSDIAG
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x0C, 12 decimal
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_SENDTRANSDIAG TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 0C00 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
Data Fields: Data Fields: aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. Entering the Transaction ID is less ambiguous. The string – "All" is NOT allowed and will result in a command error.	
Operation:	This command is used to get diagnostic data on a particular transaction. There are two types of diagnostic data sent in the packet, engine diagnostic and application diagnostic data. If transaction is not active, the engine diagnostic data will not be valid. If the transaction is not found, the command error counter will increment and error event CF_SND_TRANS_ERR_EID will be sent.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Debug event CF_SND_TRANS_CMD_EID will be sent. Use EVS command SCX_CPU1_EVS_ENAAPPEVTTYPE APPLICATION="CF" EVENT_TYPE=DEBUG to unfilter CF debug events.

B.14 SET POLL PARAMETER

Command Name:	SET POLL DIRECTORY PARAMETER
Mnemonic:	/SCX_CPU1_CF_SETPOLLPARAM
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x0D, 13 decimal
Data Field Length:	152
Syntax Example:	/SCX_CPU1_CF_ SETPOLLPARAM Chan_0 POLLDIR_2 CLASS_2 PRIORITY=9 DELETE_FILE PEERENTITYID="0.23" SRCPATH="/sdr/gmi/hk/" DSTPATH= "cffiles/"
CCSDS Format:	18B3 C000 0099 0D00 aabb ccdd eeff ffff gggg gggg gggg gggg gggg gggg gggg gggg hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh hhhh iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii iiii
Data Field: Union aa – Chan_0(0), Chan_1(1)	
Data Field: Union bb – PollDir_0(0), PollDir_1(1)….PollDir_7(7)	
Data Field: Union cc – Class_1(1), Class_2(2)	
Data Field: dd – Priority – 00 (highest priority) – 0xff (lowest priority)	
Data Field: Union ee – Delete_File(0), Keep_File(1) – Action taken after a successful transfer	
Data Field: ff – spare byte	
Data Field: gg – PeerEntityID - 16 byte string – Entity ID of ground engine to receive file	
Data Field: hh – SrcPath - 64 byte field – string – absolute path of directory to playback. String must end with forward slash '/' character.	
Data Field: ii – DstPath - 64 byte field – string – absolute path or path relative to gnd engine default path. Example "cffiles/" string will have file stored in $WORK/image/cffiles directory. String must end with forward slash '/' character.	
Operation:	This command is used to change one or more polling directory configuration parameters. All eight parameter values given in this command will be written to the CF configuration table. Unchanged parameter values must contain the current setting.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments. Debug Event CF_SET_POLL_PARAM1_EID will be sent in response to the command. Use EVS command SCX_CPU1_EVS_ENAAPPEVTTYPE APPLICATION="CF" EVENT_TYPE=DEBUG to unfilter CF debug events.

B.15 SEND CONFIGURATION PACKET

Command Name:	SEND CONFIGURATION PACKET
Mnemonic:	/SCX_CPU1_CF_SENDCFG
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x0E, 14 decimal
Data Field Length:	0
CCSDS Format:	18B3 C000 0001 0E00
Data Fields: None.	
Operation:	Upon receipt of this command, CF will build and send the telemetry packet named "CF Configuration Packet" defined in the appendix. This command is used to view the compile-time configuration parameters from the platform configuration header file (cf_platform_cfg.h) and the engine timer values. To see run-time configuration parameters see Appendix A.4 or (in the case of changed parameters) dump the CF configuration table.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution counter increments.

B.16 WRITE QUEUE INFO TO FILE

Command Name:	WRITE QUEUE INFO TO FILE
Mnemonic:	/SCX_CPU1_CF_WRITEQUE2FILE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x0F, 15 decimal
Data Field Length:	68
Syntax Example:	/SCX_CPU1_CF_WRITEQUE2FILE QTYPE=1 CHAN=0 QUE=1 FILENAME="/ram/queinfo.txt
CCSDS Format:	18B3 C000 0045 0F00 aabb ccdd eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee eeee
Data Field: aa – QTYPE – 1 byte field - 1=up/incoming, 2=downlink/outgoing	
Data Field: bb – CHAN – 1 byte field – 0=CF Chan 0, 1=CF Chan 1 This parameter is ignored by CF when QTYPE=1(uplink).	
Data Field: cc – QUE – 1 byte field - 0=Pending Queue, 1=Active Queue, 2=History Queue NOTE: A value of 0(pending queue) is not valid when QTYPE=1(uplink)	
Data Field: dd – Spare – 1 byte field	
Data Fields: ee – Filename - 64 byte string – Filename (including path) specifies the name of the file that will receive the queue data. If this parameter contains a NULL string, the CF application will use the default filename defined by CF_DEFAULT_QUEUE_INFO_FILENAME in the CF platform configuration file.	
Operation:	This command is used to write the entries in the specified queue, to a file. Further action is required (download file) to view contents of queue on Asist page.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Debug event CF_SND_Q_INFO_EID is sent.

B.17 ENABLE DEQUEUE

Command Name:	ENABLE DEQUEUE
Mnemonic:	/SCX_CPU1_CF_ENADEQUE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x10, 16 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_ENADEQUE Chan_0
CCSDS Format:	18B3 C000 0005 1000 aabb bbbb
Data Field: Union aa – Chan_0(0), Chan_1(1)	
Data Field: bb – spare	
Operation:	This command is used to enable dequeuing of the pending queue on the specified channel.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Dequeue State in HK tlm or dump CF Configuration table to verify.

B.18 DISABLE DEQUEUE

Command Name:	DISABLE DEQUEUE
Mnemonic:	/SCX_CPU1_CF_DISDEQUE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x11, 17 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_DISDEQUE Chan_0
CCSDS Format:	18B3 C000 0005 1100 aabb bbbb
Data Field: Union aa – Chan_0(0), Chan_1(1)	
Data Field: bb – spare	
Operation:	This command is used to disable dequeuing of the pending queue on the specified channel.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Dequeue State in HK tlm or dump CF Configuration table to verify.

B.19 ENABLE POLL DIRECTORY CHECKING

Command Name:	ENABLE POLL DIRECTORY CHECKING
Mnemonic:	/SCX_CPU1_CF_ENAPOLL
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x12, 18 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_ENAPOLL Chan_1 POLLDIR_ALL
CCSDS Format:	18B3 C000 0005 1200 aabb cccc
Data Field: Union aa – Chan_0(0), Chan_1(1)	
Data Field: Union bb – PollDir_0(0), PollDir_1(1)….PollDir_7(7), PollDir_All(8)	
Data Field: cc – spare	
Operation:	This command is used to enable one or all polling directories on the specified channel.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. The corresponding polling enable bit will be a value of one. The polling enable status in telemetry is displayed as a 2 digit hex number. Each bit corresponds to one polling directory. Polling directory zero corresponds to the least significant bit. For polling directory numbers refer to Appendix A.4 or dump the CF Configuration Table.

B.20 DISABLE POLL DIRECTORY CHECKING

Command Name:	DISABLE POLL DIRECTORY CHECKING
Mnemonic:	/SCX_CPU1_CF_DISPOLL
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x13, 19 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_DISPOLL Chan_1 POLLDIR_ALL
CCSDS Format:	18B3 C000 0005 1300 aabb cccc
Data Field: Union aa – Chan_0(0), Chan_1(1)	
Data Field: Union bb – PollDir_0(0), PollDir_1(1)….PollDir_7(7), PollDir_All(8)	
Data Field: cc – spare	
Operation:	This command is used to disable one or all polling directories on the specified channel.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. The corresponding polling enable bit will be a value of zero. The polling enable status in telemetry is displayed as a 2 digit hex number. Each bit corresponds to one polling directory. Polling directory zero corresponds to the least significant bit. For polling directory numbers refer to Appendix A.4 or dump the CF Configuration Table.

B.21 DELETE QUEUE NODE

Command Name:	DELETE QUEUE NODE
Mnemonic:	/SCX_CPU1_CF_DEQUENODE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x14, 20 decimal
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_ DEQUENODE TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 1400 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa

Data Fields: aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. When removing nodes from the pending queues, the filename format should be used because transaction numbers on all entries are not yet assigned and default to zero (ex. Transaction IDs for all pending queue entries are 0.24_0). The string – "All" is not valid for this command. Use the purge queue command to remove all nodes on a particular queue.

Operation:	This command is used to remove a node from a queue. A queue node is also known as a queue entry. This command is most often used to remove queue entries for files that are pending. This command may also be used to free memory by removing queue entries from the history queue. In the unlikely event that a transaction gets stuck on an active queue, this command may be used to remove the node from the active queue. This is not an expected scenario. Using this command to remove a queue entry from an active queue may result in an adverse affect of unknown consequences. The CF application has protection against accidentally removing a queue node from an active queue. If the transaction specified in the command is found to be on the active queue, a critical event message will be sent (CF_DEQ_NODE_ERR2_EID for uplink transactions or CF_DEQ_NODE_ERR3_EID for playback transactions), indicating that the transaction is active and the command must be sent again to take affect.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Verify telemetry counter XXXQFileCount decrements, where XXX is Pending, Active or History

B.22 PURGE QUEUE

Command Name:	PURGE QUEUE
Mnemonic:	/SCX_CPU1_CF_PURGEQUEUE
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x15, 21 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_PURGEQUEUE QTYPE=2 Chan_0 QUE=2
CCSDS Format:	18B3 C000 0005 1500 aabb ccdd
Data Field: aa – QTYPE – 1 byte field - 1=up/incoming, 2=downlink/outgoing	
Data Field: Union bb – Chan_0(0), Chan_1(1) This parameter is ignored by CF when QTYPE=1(uplink).	
Data Field: cc – QUE – 1 byte field - 0=Pending Queue, 2=History Queue NOTE: A value of 0 (pending queue) is not valid when QTYPE=1(uplink) NOTE: A value of 1 (active queue) is not allowed, cannot purge an active queue.	
Data Field: dd – Spare – 1 byte field	
Operation:	This command is used to remove all entries in the specified queue. CF does not allow purging an active queue.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Info event CF_PURGEQ1_EID (for incoming history queue) or CF_PURGEQ2_EID (for outgoing queue) is sent.

B.23 WRITE ACTIVE TRANSACTIONS TO FILE

Command Name:	WRITE ACTIVE TRANSACTIONS TO FILE
Mnemonic:	/SCX_CPU1_CF_WRITEACTIVETRANS
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x16, 22 decimal
Data Field Length:	65
Syntax Example:	/SCX_CPU1_CF_ WRITEACTIVETRANS ALL /ram/activetransactions.txt
CCSDS Format:	18B3 C000 0042 1600 aabb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bbbb bb
Data Fields: Union aa – ALL(0), INCOMING(1), OUTGOING(2)	
Data Fields: bb – Filename - 64 byte string – Filename (including path) specifies the name of the file that will receive the data. If this parameter contains a NULL string, the CF application will use the default filename defined by CF_DEFAULT_QUEUE_INFO_FILENAME in the CF platform configuration file.	
Operation:	This command will write the contents of one or more active queues to a file. CF has an active queue for each output channel and one active queue for incoming transactions. After the file is written successfully, steps must be taken to transfer the file to the ground system so that the information can be viewed on the page.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Debug event CF_WRACT_TRANS_EID is sent.

B.24 KICKSTART

Command Name:	KICKSTART
Mnemonic:	/SCX_CPU1_CF_KICKSTART
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x17, 23 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_KICKSTART Chan_1
CCSDS Format:	18B3 C000 0005 1700 aabb bbbb
Data Fields: Union aa – Chan_0(0), Chan_1(1)	
Data Fields: Union bb – spare	
Operation:	This command can be used in the unlikely event that PDUs stop flowing on an output channel that is enabled and has files waiting to be sent. Once a transaction begins and other files are pending, the CF design attempts to maintain a steady flow of outgoing PDUs on each channel. The CF application will start the next file on the pending queue, immediately after the EOF PDU of the current transaction is sent out. In the unlikely scenario that the engine does not output the EOF PDU of a transaction, the next file on the pending queue will not begin. CAUTION: This command is not needed under normal conditions and should be used only when cancel/abandon/resume/channel-enable commands fail to start the next pending file. NOTE: Suspending an outgoing transaction before EOF is sent, will pause the flow of PDUs on that channel. If a user wishes to stop the current transaction (before the EOF is sent) and still allow the next pending file to begin, the current transaction should be cancelled (or abandoned) in lieu of being suspended. In most cases, canceling is a better option than abandoning a transaction. WARNING: It is not recommended that this command be used when a suspended transaction has paused the flow of PDUs on a channel. Instead, resume the suspended transaction, then cancel or abandon that transaction.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Debug event CF_KICKSTART_CMD_EID is sent.

B.25 QUICK STATUS

Command Name:	QUICK STATUS
Mnemonic:	/SCX_CPU1_CF_QUICKSTATUS
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x18, 24 decimal
Data Field Length:	64
Syntax Example:	/SCX_CPU1_CF_QUICKSTATUS TRANSIDORFILENAME="0.24_4"
CCSDS Format:	18B3 C000 0041 1800 aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa aaaa
Data Fields: aa – TransIdOrFilename - 64 byte field – string – Transaction ID (0.24_4) or Filename (including path) of the transaction to suspend. The CF application assumes a filename is given if the first character is a forward slash, otherwise the string is assumed to be transaction ID. Entering the Transaction ID is less ambiguous. The string – "All" is not allowed.	
Operation:	This command is used to display high level status of the specified transaction in an event message. High level status may include Pending, Active, Success, Failed. If a file transaction has failed, the condition code will also be displayed indicating the reason for failure. The status will also show if a transaction has been suspended.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. Informational event CF_QUICK_CMD_EID is sent indicating transaction ID, filename and status.

B.26 SEMAPHORE GIVE/TAKE COMMAND

Command Name:	Semaphore Give/Take Command
Mnemonic:	/SCX_CPU1_CF_ CHANSEMACTION
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x19, 25 decimal
Data Field Length:	2
Syntax Example:	/SCX_CPU1_CF_ CHANSEMACTION Chan_0 GIVE
CCSDS Format:	18B3 C000 0005 1900 aabb
Data Field: Union aa – Chan_0(0), Chan_1(1)	
Data Field: bb – GiveOrTake – 1byte – GiveSemaphore(0), TakeSemaphore(1)	
Operation:	This command is used to adjust the handshake semaphore in the unexpected case that the semaphore value lost or gained a count when viewed during idle time. Typically the Telemetry Output (TO) application will create the semaphore with an initial number of 'gives' equal to the depth of TO's input pipe. The semaphore value represents the number of empty buffers on TO's pipe. This value can get askew for example, if CF cannot deliver a PDU after taking the semaphore. In this case the user would be able to send this command to 'give' the semaphore to correct the value. This command will have no affect if throttling is not used on the specified channel. CAUTION: This command is not needed under normal conditions and should be used only when it is known that the semaphore value does not equal the expected value when the throttled channel has no active transactions.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. SCX_CPU1_CF_DOWNLINKCHAN_0_SEMVALUE will adjust by one count.

B.27 ENABLE/DISABLE AUTO SUSPEND

Command Name:	ENABLE/DISABLE AUTO SUSPEND
Mnemonic:	/SCX_CPU1_CF_ENADISAUTOSUSPEND
Message ID:	18B3
CCSDS APID:	0xB3, 179 decimal
Function Code:	0x1A, 26 decimal
Data Field Length:	4
Syntax Example:	/SCX_CPU1_CF_ ENADISAUTOSUSPEND ENABLE
CCSDS Format:	18B3 C000 0005 1A00 aaaa
Data Fields: Union aa – Disable(0), Enable(1)	
Operation:	This command is used to enable or disable auto suspend mode. When Auto suspend is enabled, CF will suspend every outgoing transaction after the EOF is sent. This is a global mode that applies to all output channels (Chan_0,Chan_1). After processor or power-on reset, the Auto suspend mode defaults to disabled.
Command Verification:	[SCX_CPU1_CF_CMDPC] Command execution increments. SCX_CPU1_CF_AutoSuspendEnFlag indicates if the mode is enabled or disabled. (i.e. 0=disabled,1=enabled)

B.28 CF Command RDL

NOTE: Some comments are incorrect or have not been updated.

```
!=============================================================================
!             Originator:      W. Moleski
!             Responsible SC:
!             Responsible CSE:
!             Rev:  Last Change: May 31 2011

!
!              SCX CPU1 CFDP Command Packet 00B3
!              ================================
!
!      Packet Application ID: 0179 (Hex '00B3')
!      Packet Title:SCX CPU1 Health and Safety App Commands
!      Packet Source:
!
!      HISTORY:
!
!  17JUL09     WFM      : Initial
!  06OCT09     WFM      : Added destination filename to playback cmd
!  07JUL10     WFM      : Added Kickstart and QuickStatus commands
!  04AUG10     WFM      : Replaced command arguments with defined unions
!  01NOV10     WFM      : Added PeerEntityID argument to the PlaybackFile,
!                         PlaybackDir, and SetPollParam commands. Also, reversed
!                         the order of the Channel and Class args to PlaybackDir
!                         so they matched the PlaybackFile command
!  18MAY11     WFM      : Updated for CFDP 2.2.0.0. Added FCTN 25 and added a
!                         spare to the Write Active Trans command
!  20MAY11     WFM      : Added FCTN 26
!
!       2011/05/31 : Created from template 'template_cmd_CF_CMD.rdl'
!       with parameters spacecraft='SCX' and processor='CPU1'.
!=============================================================================
!
#include "osconfig.h"
#include "cfe_mission_cfg.h"
#include "cf_platform_cfg.h"
#include "cf_defs.h"

! Defines to create command parameter unions
!
#define SCX_CPU1_Class     UNION CF_Class \
       UB Class_1 STATIC,DEFAULT=1,DESC="Class 1 - No Feedback" \
       UB Class_2 STATIC,DEFAULT=2,DESC="Class 2 - With Feedback" \
     END
#define SCX_CPU1_Chan     UNION CF_Channel \
       UB Chan_0 STATIC,DEFAULT=0,DESC="Channel 0" \
       UB Chan_1 STATIC,DEFAULT=1,DESC="Channel 1" \
     END
#define SCX_CPU1_Preserve UNION CF_Preserve \
       UB Delete_File STATIC,DEFAULT=0,DESC="Delete file(s) upon successful transaction" \
       UB Keep_File   STATIC,DEFAULT=1,DESC="Keep file(s) upon successful transaction" \
     END
#define SCX_CPU1_PollDirNoAll UNION CF_PollDirNoAll \
       UB PollDir_0 STATIC,DEFAULT=0,DESC="Directory 0" \
```

```
      UB PollDir_1 STATIC,DEFAULT=1,DESC="Directory 1" \
      UB PollDir_2 STATIC,DEFAULT=2,DESC="Directory 2" \
      UB PollDir_3 STATIC,DEFAULT=3,DESC="Directory 3" \
      UB PollDir_4 STATIC,DEFAULT=4,DESC="Directory 4" \
      UB PollDir_5 STATIC,DEFAULT=5,DESC="Directory 5" \
      UB PollDir_6 STATIC,DEFAULT=6,DESC="Directory 6" \
      UB PollDir_7 STATIC,DEFAULT=7,DESC="Directory 7" \
    END
#define SCX_CPU1_PollDirAll UNION CF_PollDirAll \
      UB PollDir_0   STATIC,DEFAULT=0,DESC="Directory 0" \
      UB PollDir_1   STATIC,DEFAULT=1,DESC="Directory 1" \
      UB PollDir_2   STATIC,DEFAULT=2,DESC="Directory 2" \
      UB PollDir_3   STATIC,DEFAULT=3,DESC="Directory 3" \
      UB PollDir_4   STATIC,DEFAULT=4,DESC="Directory 4" \
      UB PollDir_5   STATIC,DEFAULT=5,DESC="Directory 5" \
      UB PollDir_6   STATIC,DEFAULT=6,DESC="Directory 6" \
      UB PollDir_7   STATIC,DEFAULT=7,DESC="Directory 7" \
      UB PollDir_All STATIC,DEFAULT=255,DESC="All Directories in use" \
    END

CLASS P00B3 APID=0179, DESC="SCX CPU1 CFDP App Commands"
!
 CMD  SCX_CPU1_CF_NOOP               FCTN=0, DESC="CF no-op command"
!
 CMDS SCX_CPU1_CF_RESETCTRS     FCTN=1, DESC="CF reset counters command"
  UNION CF_ResetValue            DESC="Counter Values to reset"
    UB All      STATIC,DEFAULT=0,DESC="All Counters"
    UB Command  STATIC,DEFAULT=1,DESC="Command Counters"
    UB Fault    STATIC,DEFAULT=2,DESC="Fault Counters"
    UB Incoming STATIC,DEFAULT=3,DESC="Incoming transaction Counters"
    UB Outgoing STATIC,DEFAULT=4,DESC="Outgoing transaction Counters"
   END
  UB spare[3]                    DESC="Spare",INVISIBLE,STATIC,DEFAULT=0
 END
!
 CMDS SCX_CPU1_CF_PlaybackFile   FCTN=2, DESC="CF Playback File command"
  SCX_CPU1_Class
  SCX_CPU1_Chan
  UB   priority                  DESC="Priority"
  SCX_CPU1_Preserve
  char PeerEntityID[CF_MAX_CFG_VALUE_CHARS] DESC="Entity ID for file to be downloaded"
  char SrcFilename[OS_MAX_PATH_LEN]   DESC="Path/Name of file to be downloaded"
  char DestFilename[OS_MAX_PATH_LEN] DESC="Path/Name of destination"
 END
!
 CMDS SCX_CPU1_CF_PlaybackDir FCTN=3, DESC="CF Playback Directory command"
  SCX_CPU1_Class
  SCX_CPU1_Chan
  UB   priority                  DESC="Priority"
  SCX_CPU1_Preserve
  char PeerEntityID[CF_MAX_CFG_VALUE_CHARS] DESC="Entity ID for files to be downloaded"
  char SrcPath[OS_MAX_PATH_LEN] DESC="Path specification of the directory whose files are to be downloaded"
  char DstPath[OS_MAX_PATH_LEN] DESC="Path specification of the destination directory on the ground"
 END
!
 CMD  SCX_CPU1_CF_Freeze          FCTN=4, DESC="CF Freeze command"
!
 CMD  SCX_CPU1_CF_Thaw               FCTN=5, DESC="CF Thaw command"
```

```
!
  CMDS  SCX_CPU1_CF_Suspend         FCTN=6, DESC="CF Suspend command"
    char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_Resume          FCTN=7, DESC="CF Resume command"
    char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_Cancel          FCTN=8, DESC="CF Cancel command"
    char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_Abandon         FCTN=9, DESC="CF Abandon command"
    char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_SetAckLimit     FCTN=10, DESC="CF Set Ack Limit command"
    CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Ack Limit string
directive",INVISIBLE,STATIC,DEFAULT="ACK_LIMIT"
    CHAR NumTries[CF_MAX_CFG_VALUE_CHARS]    DESC="number of tries"
  END
!
  CMDS  SCX_CPU1_CF_SetAckTimeout    FCTN=10, DESC="CF Set Ack Timeout command"
    CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Ack timout string
directive",INVISIBLE,STATIC,DEFAULT="ACK_TIMEOUT"
    CHAR NumSeconds[CF_MAX_CFG_VALUE_CHARS]       DESC="number of seconds"
  END
!
  CMDS  SCX_CPU1_CF_SetInactivTimeout FCTN=10, DESC="CF Set Inactivity Timeout command"
    CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Inactivity timout string
directive",INVISIBLE,STATIC,DEFAULT="INACTIVITY_TIMEOUT"
    CHAR NumSeconds[CF_MAX_CFG_VALUE_CHARS]       DESC="number of seconds"
  END
!
  CMDS  SCX_CPU1_CF_SetNakLimit     FCTN=10, DESC="CF Set Nak Limit command"
    CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Nak Limit string
directive",INVISIBLE,STATIC,DEFAULT="NAK_LIMIT"
    CHAR NumTries[CF_MAX_CFG_VALUE_CHARS]    DESC="number of tries"
  END
!
  CMDS  SCX_CPU1_CF_SetNakTimeout    FCTN=10, DESC="CF Set Nak Timeout command"
    CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Nak timout string
directive",INVISIBLE,STATIC,DEFAULT="NAK_TIMEOUT"
    CHAR NumSeconds[CF_MAX_CFG_VALUE_CHARS]       DESC="number of seconds"
  END
!
  CMDS  SCX_CPU1_CF_SetSaveIncompFiles FCTN=10, DESC="CF Save Incomplete Files command"
    CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Save incomplete files string
directive",INVISIBLE,STATIC,DEFAULT="SAVE_INCOMPLETE_FILES"
    CHAR YesorNo[CF_MAX_CFG_VALUE_CHARS]     DESC="Yes or No"
  END
!
  CMDS  SCX_CPU1_CF_SetOutgoingSize FCTN=10, DESC="CF Set Outgoing File Chunk Size command"
```

```
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Outgoing File Chunk Size string
directive",INVISIBLE,STATIC,DEFAULT="OUTGOING_FILE_CHUNK_SIZE"
   CHAR NumBytes[CF_MAX_CFG_VALUE_CHARS]   DESC="Byte size of the PDU"
  END
!
  CMDS  SCX_CPU1_CF_SetInvalidParam  FCTN=10, DESC="CF Save Incomplete Files command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Invalid string
directive",INVISIBLE,STATIC,DEFAULT="INVALID_NAME"
   CHAR Value[CF_MAX_CFG_VALUE_CHARS] DESC="Dummy Value",INVISIBLE,STATIC,DEFAULT="30"
  END
!
  CMDS  SCX_CPU1_CF_GetAckLimit     FCTN=11, DESC="CF Get Ack Limit command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Ack Limit string
directive",INVISIBLE,STATIC,DEFAULT="ACK_LIMIT"
  END
!
  CMDS  SCX_CPU1_CF_GetAckTimeout    FCTN=11, DESC="CF Get Ack Timeout command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Ack timout string
directive",INVISIBLE,STATIC,DEFAULT="ACK_TIMEOUT"
  END
!
  CMDS  SCX_CPU1_CF_GetInactivTimeout FCTN=11, DESC="CF Get Inactivity Timeout command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Inactivity timout string
directive",INVISIBLE,STATIC,DEFAULT="INACTIVITY_TIMEOUT"
  END
!
  CMDS  SCX_CPU1_CF_GetNakLimit     FCTN=11, DESC="CF Get Nak Limit command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Nak Limit string
directive",INVISIBLE,STATIC,DEFAULT="NAK_LIMIT"
  END
!
  CMDS  SCX_CPU1_CF_GetNakTimeout    FCTN=11, DESC="CF Get Nak Timeout command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Nak timout string
directive",INVISIBLE,STATIC,DEFAULT="NAK_TIMEOUT"
  END
!
  CMDS  SCX_CPU1_CF_GetSaveIncompFiles FCTN=11, DESC="CF Save Incomplete Files command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Save incomplete files string
directive",INVISIBLE,STATIC,DEFAULT="SAVE_INCOMPLETE_FILES"
  END
!
  CMDS  SCX_CPU1_CF_GetOutgoingSize FCTN=11, DESC="CF Get Outgoing File Chunk Sizecommand"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Save incomplete files string
directive",INVISIBLE,STATIC,DEFAULT="OUTGOING_FILE_CHUNK_SIZE"
  END
!
  CMDS  SCX_CPU1_CF_GetInvalidParam FCTN=11, DESC="CF Save Incomplete Files command"
   CHAR Param[CF_MAX_CFG_PARAM_CHARS]        DESC="Invalid string
directive",INVISIBLE,STATIC,DEFAULT="INVALID_NAME"
  END
!
  CMDS  SCX_CPU1_CF_SendTransDiag    FCTN=12, DESC="CF Send Transaction Diagnostics command"
   char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_SetPollParam     FCTN=13, DESC="CF Set Polling Directory Parameter command"
   SCX_CPU1_Chan
```

```
   SCX_CPU1_PollDirNoAll
   SCX_CPU1_Class
   UB   priority    DESC="Priority"
   SCX_CPU1_Preserve
   UB   spare[3]                DESC="Spare",INVISIBLE,STATIC,DEFAULT=0
   char PeerEntityID[CF_MAX_CFG_VALUE_CHARS] DESC="Entity ID for the directory"
   char  SrcPath[OS_MAX_PATH_LEN] DESC="Path to poll"
   char  DstPath[OS_MAX_PATH_LEN] DESC="Path to store files on ground"
  END
!
  CMD  SCX_CPU1_CF_SendCfg          FCTN=14, DESC="CF Send Configuration Parameters command"
!
  CMDS  SCX_CPU1_CF_WritePB0PendInfo FCTN=15, DESC="CF Write Playback Channel 0 Pending Queue Info
command"
   UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=2
   UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=0
   UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=0
   UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
   CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WritePB0ActvInfo FCTN=15, DESC="CF Write Playback Channel 0 Active Queue Info
command"
   UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=2
   UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=0
   UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=1
   UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
   CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WritePB0HistInfo FCTN=15, DESC="CF Write Playback Channel 0 History Queue Info
command"
   UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=2
   UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=0
   UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=2
   UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
   CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WritePB1PendInfo FCTN=15, DESC="CF Write Playback Channel 1 Pending Queue Info
command"
   UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=2
   UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=1
   UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=0
   UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
   CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WritePB1ActvInfo FCTN=15, DESC="CF Write Playback Channel 1 Active Queue Info
command"
   UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=2
   UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=1
   UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=1
   UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
   CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
```

```
  CMDS  SCX_CPU1_CF_WritePB1HistInfo FCTN=15, DESC="CF Write Playback Channel 1 History Queue Info
command"
    UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=2
    UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=1
    UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=2
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
    CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WriteUpActvInfo FCTN=15, DESC="CF Write Uplink Active Queue Info command"
    UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=1
    UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=%XFF
    UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=1
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
    CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WriteUpHistInfo FCTN=15, DESC="CF Write Uplink History Queue Info command"
    UB qtype        DESC="Queue Type (up=1,down=2)",INVISIBLE,STATIC,DEFAULT=1
    UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=%XFF
    UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=2
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
    CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_WriteQue2File FCTN=15, DESC="CF Write Queue Info to File command"
    UB qtype        DESC="Queue Type (up=1,down=2)"
    UB chan DESC="Channel number"
    UB que   DESC="Queue number"
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
    CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the Queue Info to"
  END
!
  CMDS  SCX_CPU1_CF_EnaDeque      FCTN=16, DESC="CF Enable Dequeue command"
    SCX_CPU1_Chan
    UB spare[3]             DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_DisDeque      FCTN=17, DESC="CF Disable Dequeue command"
    SCX_CPU1_Chan
    UB spare[3]             DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_EnaPoll    FCTN=18, DESC="CF Enable Directory Polling command"
    SCX_CPU1_Chan
    SCX_CPU1_PollDirAll
    UB spare[2]             DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_DisPoll       FCTN=19, DESC="CF Disable Directory Polling command"
    SCX_CPU1_Chan
    SCX_CPU1_PollDirAll
    UB spare[2]             DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
!
  CMDS  SCX_CPU1_CF_DequeNode      FCTN=20, DESC="CF Deque Node command"
```

```
   char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_PurgePendingQue FCTN=21, DESC="CF Purge Pending Queue command"
    UB qtype         DESC="Queue Type",INVISIBLE,STATIC,DEFAULT=2
    SCX_CPU1_Chan
    UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=0
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_PurgeInHistoryQue FCTN=21, DESC="CF Purge Incoming History Queue command"
    UB qtype         DESC="Queue Type",INVISIBLE,STATIC,DEFAULT=1
    UB chan DESC="Channel number",INVISIBLE,STATIC,DEFAULT=%XFF
    UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=2
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_PurgeOutHistoryQue FCTN=21, DESC="CF Purge Outgoing History Queue command"
    UB qtype         DESC="Queue Type",INVISIBLE,STATIC,DEFAULT=2
    SCX_CPU1_Chan
    UB que   DESC="Queue number",INVISIBLE,STATIC,DEFAULT=2
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_WriteActiveTrans FCTN=22, DESC="CF Write Active Transaction Information command"
    UNION qtype  DESC="Queue Type to write"
      UB All      STATIC,DEFAULT=0,DESC="Incoming and Outgoing Active Queues"
      UB Incoming STATIC,DEFAULT=1,DESC="Incoming Active Queue"
      UB Outgoing STATIC,DEFAULT=2,DESC="Outgoing Active Queue"
    END
    UB spare DESC="",INVISIBLE,STATIC,DEFAULT=0
    CHAR FileName[OS_MAX_PATH_LEN] DESC="Name of the file to write the information to"
  END
!
  CMDS  SCX_CPU1_CF_KickStart FCTN=23, DESC="CF Kick Start command"
    SCX_CPU1_Chan
    UB spare[3]     DESC="",INVISIBLE,STATIC,DEFAULT=0
  END
!
  CMDS  SCX_CPU1_CF_QuickStatus FCTN=24, DESC="CF Quick Status command"
    char TransIdorFilename[OS_MAX_PATH_LEN] DESC="String - Transaction Id (i.e.: 0.24_5) or Filename (starts
with /)"
  END
!
  CMDS  SCX_CPU1_CF_ChanSemAction FCTN=25, DESC="CF Give or Take Semaphore command"
    SCX_CPU1_Chan
    UNION semAction   DESC="The action to take on the semaphore"
      UB Give STATIC,DEFAULT=0,DESC="Give Semaphore back"
      UB Take STATIC,DEFAULT=1,DESC="Take Semaphore"
    END
  END
!
  CMDS  SCX_CPU1_CF_AutoSuspend FCTN=26, DESC="CF Auto Suspend command"
    UNION suspendAction DESC="The action to take"
      ULI Disable STATIC,DEFAULT=0,DESC="Disable"
      ULI Enable STATIC,DEFAULT=1,DESC="Enable"
    END
```

```
  END
!
END      !END PACKET
!
!=============================================================================
```

Appendix C - Telemetry

C.1 CF Housekeeping Telemetry Packet

The CF application sends a housekeeping telemetry packet to the software bus every 4 seconds.
The Asterisk (*) indicates telemetry points that show available CF resources. If these values reach the suggested yellow limit settings, it may indicate CF configuration parameters need to be adjusted.

Name	Size in Bytes	Data Type	Description
CCSDS Primary Header	6	struct	Primary Header
Secondary Header	6	struct	Most Significant 4 bytes – Seconds, LS 2 bytes - Subseconds
Command Counter	2	uint16	Number of Commands Executed successfully
Command Error Counter	2	uint16	Number of Command errors
Wake Up Count For File Processing	4	uint32	Number of Wakeup commands received
Engine Cycle Count	4	uint32	Number of times the engine is cycled
Memory In Use	4	uint32	Bytes of memory for queue entries currently in use
Peak Memory In Use	4	uint32	Highest value since reset
Low Memory Mark *	4	uint32	Num Bytes of free memory for Queue entries. Suggested Yellow Lim <40K
Max Memory Needed	4	uint32	Mem needed when all queues are full (not including mem pool overhead)
Memory Allocated	4	uint32	The platform configuration parameter that dictates the amount of memory
Buffer Pool Handle	4	uint32	Memory Pool Handle if stats are needed through CFE ES.
Queue Nodes Allocated	4	uint32	Number of Queue Nodes allocated from the memory pool
Queue Nodes Deallocated	4	uint32	Number of Queue Nodes returned to the memory pool heap
PDUs Received	4	uint32	Number of PDUs received
PDUs Rejected	4	uint32	Number of PDUs rejected (PDU too large for buffer or wrong hdr length)
Total Transactions In Progress *	4	uint32	Should not get close to platform cfg setting, Suggested Yellow Lim >380
Total Failed Transactions	4	uint32	Includes cancelled, abandoned or transactions that timeout.
Total Abandon Transactions	4	uint32	Includes uplink and downlink transactions
Total Successful Transactions	4	uint32	Includes uplink and downlink transactions
Total Completed Transactions	4	uint32	Should equal sum of total failed and total successful
Last Failed Transaction	16	string	May be uplink or downlink, shows transaction ID
Auto Suspend Enable Flag	4	uint32	Auto Suspend mode enabled (1) or disabled (0)
Auto Suspend Buffers Free *	4	uint32	Number of Free buffers in Auto Suspend mode. Suggested Yellow Lim <5
Positive Ack Limit Errors (flight side)	1	uint8	After sending EOF, CF did not receive Ack-EOF within timeout
File Store Rejection Errors(flight side)	1	uint8	Onboard File System returned error to CF
File Checksum Errors (flight side)	1	uint8	Flight engine reports file checksum error
File Size Errors (flight side)	1	uint8	Discrepancy btwn file size given by file sys and flght eng calculated size
Nak Limit Errors (flight side)	1	uint8	Applies only to class 2 uplink, if flight did not rcv pkt in response to Nak
Inactivity Timeout Errors (flight side)	1	uint8	Uplink only, if CF stopped receiving incoming PDUs during transaction
Suspend Request Counter (flight side)	1	uint8	Number of transactions suspended,(does not work) See known problems #2
Cancel Request Counter (flight side)	1	uint8	Number of transactions cancelled
Flight Entity ID	16	string	flight entity id reported by engine
Flags	4	uint32	Bit 0, 0=Thawed, 1=Frozen, set by CF. (Any partners frozen more accurate)
Machines Allocated	4	uint32	transactions in progress reported by engine
Machines Deallocated	4	uint32	completed transactions reported by engine
Are Any Partners Frozen	1	uint8	Should read…Is Flight Engine Frozen, 1=yes, 0=no, set by engine
spare	3	uint8	
how many senders	4	uint32	Number of active outgoing transactions reported by engine
how many receivers	4	uint32	Number of active incoming transactions reported by engine
how many frozen	4	uint32	Number of frozen transactions reported by engine
how many suspended	4	uint32	Number of suspended transactions reported by engine
total file sent	4	uint32	total files sent reported by engine
total file received	4	uint32	total files received reported by engine
total unsuccessful senders	4	uint32	total unsuccessful outgoing transactions reported by engine

Name	Size in Bytes	Data Type	Description
total unsuccessful receivers	4	uint32	total unsuccessful incoming transactions reported by engine
Metadata PDUs received	4	uint32	Self-Explanatory
Uplink Active Queue File Count	4	uint32	Self-Explanatory
Uplink Success Counter	4	uint32	successful incoming transactions reported by CF
Uplink Failed counter	4	uint32	unsuccessful incoming transactions reported by CF
Last File Uplinked	64	String	path/filename
Out Chan 0 PDUs Sent	4	uint32	Self-Explanatory
Out Chan 0 Files Sent	4	uint32	Self-Explanatory
Out Chan 0 Success Counter	4	uint32	Successful Transactions on Out Chan 0
Out Chan 0 Failed Counter	4	uint32	Failed Transactions on Out Chan 0
Out Chan 0 Pending Queue File Count	4	uint32	Self-Explanatory
Out Chan 0 Active Queue File Count	4	uint32	Self-Explanatory
Out Chan 0 History Queue File Count	4	uint32	Self-Explanatory
Out Chan 0 Flags	4	uint32	Bit 0: Pending Queue Dequeue State, 0=disabled, 1= enabled
			Bit 1: Blasting in Progress, 0=no, 1=yes. (Continuous PDU stream in prog)
			Bit 2-9: Polling enable state, 0=disabled,1=enabled, poll dir 0 = Bit 2
Out Chan 0 Red Light Counter	4	uint32	Number of times TO has held off CF from sending PDU to software bus
Out Chan 0 Green Light Counter	4	uint32	Number of times TO has granted CF permission to send PDU to SB
Out Chan 0 Polling Directories Checked Cnt	4	uint32	Number of times polling directories have been checked
Out Chan 0 Pending Queue Checked Cnt	4	uint32	Number of times pending queue checked (applies only when transactions are not in progress)
Out Chan 0 Semaphore Value	4	uint32	CF-TO Counting Semaphore value, number of empty buffers on TO pipe
Out Chan 1 PDUs Sent	4	uint32	Self-Explanatory
Out Chan 1 Files Sent	4	uint32	Self-Explanatory
Out Chan 1 Success Counter	4	uint32	Successful Transactions on Out Chan 1
Out Chan 1 Failed Counter	4	uint32	Failed Transactions on Out Chan 1
Out Chan 1 Pending Queue File Count	4	uint32	Self-Explanatory
Out Chan 1 Active Queue File Count	4	uint32	Self-Explanatory
Out Chan 1 History Queue File Count	4	uint32	Self-Explanatory
Out Chan 1 Flags	4	uint32	Bit 0: Pending Queue Dequeue State, 0=disabled, 1= enabled
			Bit 1: Blasting in Progress, 0=no, 1=yes. (Continuous PDU stream in prog)
			Bit 2-9: Polling enable state, 0=disabled,1=enabled, poll dir index 0 = Bit 2
Out Chan 1 Red Light Counter	4	uint32	Number of times TO has held off CF from sending PDU to software bus
Out Chan 1 Green Light Counter	4	uint32	Number of times TO has granted CF permission to send PDU to SB
Out Chan 1 Polling Directories Checked Cnt	4	uint32	Number of times polling directories have been checked
Out Chan 1 Pending Queue Checked Cnt	4	uint32	Number of times pending queue checked (applies only when transactions are not in progress)
Out Chan 1 Semaphore Value	4	uint32	CF-TO Counting Semaphore value, number of empty buffers on TO pipe

C.2 CF Housekeeping RDL

NOTE: Some comments are incorrect or have not been updated.

```
!=============================================================================
!
!             Originator: W. Moleski
!             Responsible SC:
!             Responsible CSE:
!             Rev:  Last Change: May 20 2011

!
!               Telemetry Packet # 0176 (dec)
!              ================================
!
!      Packet Application ID: 0176 (Hex '00B0')
!      Packet Title:  SCX CPU1 CF Telemetry Data Packet
!      Packet Length:  ?? Bytes (Including 12 Bytes Of Header)
!       Collect            Frequency:         SEC
!
!      REFERENCES:
!
!      NOTES:
!
!      HISTORY:
!
!  05NOV09     WFM      : Initial
!  18MAY11     WFM      : Updated for CFDP 2.2.0.0. Added SemValue to downlink
!                         telemetry struct as well as created a union for the
!                         downlink flags
!
!       2011/05/20 : Created from template 'template_tlm_CF_HK_TLM.rdl'
!       with parameters spacecraft='SCX' and processor='CPU1'.
!=============================================================================
!
#include "osconfig.h"
#include "cf_platform_cfg.h"
!
TYPES
  RECORD SCX_CPU1_CF_DownlinkType    DESC="Downlink Telemetry Record"
     ULI  PDUsSent       DESC=""
     ULI  FilesSent      DESC=""
     ULI  GoodDownlinkCnt  DESC=""
     ULI  BadDownlinkCnt   DESC=""
     ULI  PendingQFileCnt  DESC=""
     ULI  ActiveQFileCnt   DESC=""
     ULI  HistoryQFileCnt  DESC=""
     UNION DownlinkFlags   DESC=""
       ULI DequeueFlag              DESC="Dequeue State Flag",mask=%x00000001,
                  DISCRETE, DRANGE=(0,1),DLABEL=("Disabled","Enabled")
       ULI DataBlastFlag      DESC="Data Blast State Flag",mask=%x00000002,
             DISCRETE, DRANGE=(0,1),DLABEL=("Not In Progress","In Progress")
       ULI AllFlags       DESC="Downlink Flags",mask=%xffffffff
     END
     ULI  RedLightCtr           DESC=""
```

```
    ULI  GreenLightCtrDESC=""
    ULI  PollDirsChkCtr       DESC=""
    ULI  PendingQChkCtr       DESC=""
    ULI  SemValue             DESC=""
  END
END_TYPES

PACKET P00B0 APID=0176, DESC="SCX CPU1 CFDP Telemetry Data Packet", STALE = 36
#include "ccsds_header.rdl"
!
  UI  SCX_CPU1_CF_CMDPC       DESC="CFDP Command Processed Counter",
                              UNITS=Counts
!
  UI  SCX_CPU1_CF_CMDEC       DESC="CFDP Command Error Counter",
                              UNITS=Counts, YH=1, DOLIMIT
!
  ULI SCX_CPU1_CF_FileWakeupCnt   DESC=""
  ULI SCX_CPU1_CF_EngnCycleCnt    DESC=""
  ULI SCX_CPU1_CF_MemInUse DESC=""
  ULI SCX_CPU1_CF_PeakMemInUse    DESC=""
  ULI SCX_CPU1_CF_LowMemMark      DESC=""
  ULI SCX_CPU1_CF_MaxMemNeeded DESC=""
  ULI SCX_CPU1_CF_MemAllocated    DESC=""
  ULI SCX_CPU1_CF_PoolHandle DESC=""
  ULI SCX_CPU1_CF_QNodesAlloc     DESC=""
  ULI SCX_CPU1_CF_QNodesDealloc   DESC=""
  ULI SCX_CPU1_CF_PDUsRcvd DESC=""
  ULI SCX_CPU1_CF_PDUsRejected    DESC=""
!
  ULI SCX_CPU1_CF_TotalInProgTrans DESC=""
  ULI SCX_CPU1_CF_TotalFailedTrans  DESC=""
  ULI SCX_CPU1_CF_TotalAbandTrans DESC=""
  ULI SCX_CPU1_CF_TotalGoodTrans   DESC=""
  ULI SCX_CPU1_CF_TotalCompleteTrans     DESC=""
  char SCX_CPU1_CF_LastFailedTrans[CF_MAX_TRANSID_CHARS]      DESC=""
!
  ULI SCX_CPU1_CF_AutoSuspendFlag DESC="", DISCRETE,DRANGE=(0,1),
                                  DLABEL=("Disabled","Enabled")
  ULI SCX_CPU1_CF_AutoSuspendLFM        DESC=""
!
  UB SCX_CPU1_CF_PosAckNumDESC=""
  UB SCX_CPU1_CF_FileStoreRejNum  DESC=""
  UB SCX_CPU1_CF_FileChecksumNum        DESC=""
  UB SCX_CPU1_CF_FileSizeNum        DESC=""
  UB SCX_CPU1_CF_NakLimitNum      DESC=""
  UB SCX_CPU1_CF_InactiveNum        DESC=""
  UB SCX_CPU1_CF_SuspendNum       DESC=""
  UB SCX_CPU1_CF_CancelNum  DESC=""
!
  CHAR SCX_CPU1_CF_FlightEntityId[CF_MAX_CFG_VALUE_CHARS]DESC=""
  UNION SCX_CPU1_CF_EngineFlags   DESC=""
    ULI SCX_CPU1_CF_FreezeFlag        DESC="Engine Freeze Flag",mask=%x00000001,
                              DISCRETE, DRANGE=(0,1),DLABEL=("Thaw","Frozen")
    ULI SCX_CPU1_CF_AllFlags    DESC="Engine Flags",mask=%xffffffff
  END
  ULI SCX_CPU1_CF_MachinesAlloc    DESC=""
  ULI SCX_CPU1_CF_MachinesDealloc DESC=""
  UB SCX_CPU1_CF_PartnersFrozen     DESC="",
```

```
                    DISCRETE, DRANGE=(0,1), DLABEL=("False","True")
  UB SCX_CPU1_CF_Spare2[3]    DESC=""
  ULI SCX_CPU1_CF_NumSenders       DESC=""
  ULI SCX_CPU1_CF_NumReceivers     DESC=""
  ULI SCX_CPU1_CF_NumFrozen DESC=""
  ULI SCX_CPU1_CF_NumSuspended    DESC=""
  ULI SCX_CPU1_CF_TotalFilesSent     DESC=""
  ULI SCX_CPU1_CF_TotalFilesRcvd    DESC=""
  ULI SCX_CPU1_CF_TotalBadSenders  DESC=""
  ULI SCX_CPU1_CF_TotalBadReceivers       DESC=""
!
  ULI SCX_CPU1_CF_MetaCount  DESC=""
  ULI SCX_CPU1_CF_ActiveQFileCnt    DESC=""
  ULI SCX_CPU1_CF_GoodUplinkCtr    DESC=""
  ULI SCX_CPU1_CF_BadUplinkCtr      DESC=""
  CHAR SCX_CPU1_CF_LastFileUplinked[OS_MAX_PATH_LEN]   DESC=""
!
  SCX_CPU1_CF_DownlinkType SCX_CPU1_CF_DownlinkChan[0 .. CF_MAX_PLAYBACK_CHANNELS-1]
        DESC="Downlink Telemetry channels"
!
!  END              !END APPEND RECORD FUNCTION
!
END
```

C.3 Transaction Status Packet

Name	Size in Bytes	Data Type	Description
CCSDS Primary Header	6	struct	Primary Header
Secondary Header	6	struct	Most Significant 4 bytes – Seconds, LS 2 bytes - Subseconds
TransLen	1	uint8	Engine status - Transaction number Length
TransVal	1	uint8	Engine status - Transaction number Value
Naks	1	uint8	Engine status
PartLen	1	uint8	Engine status
PartVal	1	uint8	Engine status
spare	2	uint8	Engine status
Flags	4	uint32	Engine status
TransNum	4	uint32	Engine status
Attempts	4	uint32	Engine status
CondCode	4	uint32	Engine status
DeliCode	4	uint32	Engine status
FdOffset	4	uint32	Engine status
FdLength	4	uint32	Engine status
Checksum	4	uint32	Engine status
FinalStat	4	uint32	Engine status
FileSize	4	uint32	Engine status
RcvdFileSize	4	uint32	Engine status
Role	4	uint32	Engine status
State	4	uint32	Engine status
StartTime	4	uint32	Engine status
SrcFile	64	string	Engine status
DstFile	64	string	Engine status
TmpFile	64	string	Engine status
Status	4	uint32	Unk(0),Success,Canceled,Aband,NoMeta,Pending,AlrdyAct,PutIssued, PutFailed, Active
CondCode	4	uint32	CF status
Priority	4	uint32	CF status

Name	Size in Bytes	Data Type	Description
Class	4	uint32	CF status
Out Channel Num	4	uint32	CF status
Source	4	uint32	CF status – Poll directory, Playback File Cmd, Playback Dir Cmd
NodeType	4	uint32	CF status – Uplink, Downlink
TransNum	4	uint32	CF status
SrcFile	64	string	CF status
DstFile	64	string	CF status
TmpFile	64	string	CF status

C.4 Configuration Packet

Name	Size in Bytes	Data Type	Description
CCSDS Primary Header	6	struct	Primary Header
Secondary Header	6	struct	Most Significant 4 bytes – Seconds, LS 2 bytes - Subseconds
Engine Cycles Per Wakeup	4	uint32	Copied from CF Configuration table
Ack Limit	4	uint32	Number of Ack Timeouts before cancel (from CF Config Table)
Ack Timeout	4	uint32	seconds (from CF Config Table)
Nak Limit	4	uint32	Number of Nak Timeouts before cancel (from CF Config Table)
Nak Timeout	4	uint32	seconds (from CF Config Table)
Inactivity Timeout	4	uint32	seconds (from CF Config Table)
Outgoing File Chunk Size	4	uint32	bytes (from CF Config Table)
Pipe Depth	4	uint32	CF Command pipe (from platform cfg file)
Max Simultaneous Transactions	4	uint32	Copied from platform Cfg File
Incoming PDU Buffer Size	4	uint32	Copied from platform Cfg File
Outgoing PDU Buffer Size	4	uint32	Copied from platform Cfg File
Num Input Channels	4	uint32	Copied from platform Cfg File
Max Playback Channel	4	uint32	Copied from platform Cfg File
Max Polling Dirs Per Channel	4	uint32	Copied from platform Cfg File
Memory Pool Bytes	4	uint32	Copied from platform Cfg File
Debug Compiled in	4	uint32	Copied from platform Cfg File
Save Incomplete Files?	8	string	(from CF Config Table)Yes or No
Pipe Name	20	string	Copied from platform Cfg File
Tmp File Prefix	64	string	Copied from platform Cfg File
Cfg Table Name	64	string	Copied from platform Cfg File
Cfg Table Filename	64	string	Copied from platform Cfg File
Default Queue Info Filename	64	string	Copied from platform Cfg File

C.5 Queue Information File

File content has a variable number of queue entries, only one queue entry is listed.

Name	Size in Bytes	Data Type	Description
File Hdr - Content Type	4	uint32	TBD
File Hdr – SubType	4	uint32	Most Significant 4 bytes – Seconds, LS 2 bytes - Subseconds
File Hdr – Length	4	uint32	TBD
File Hdr – Spacecraft ID	4	uint32	TBD
File Hdr – Processor ID	4	uint32	TBD
File Hdr – Application ID	4	uint32	TBD
File Hdr – Time, Seconds	4	uint32	File Creation time
File Hdr – Time, Subseconds	4	uint32	File Creation time
File Hdr - Description	32	string	
Entry 1, Transaction Status	4	uint32	Unk(0),Success,Canceled,Aband,NoMeta,Pending,AlrdyAct,PutIssued, PutFailed, Active
Entry 1, Transaction Number	4	uint32	Assigned by engine
Entry 1, Source Entity ID	16	string	TBD
Entry 1, Source Filename	64	string	TBD

Appendix D - Event Messages

#define CF_INIT_EID 1
CFE_EVS_INFORMATION, "CF Initialized. Version %d.%d.%d.%d"

#define CF_CC_ERR_EID 2
CFE_EVS_ERROR, "Cmd Msg with Invalid command code Rcvd -- ID = 0x%04X, CC = %d"

#define CF_MID_ERR_EID 3
CFE_EVS_ERROR, "Unexpected Msg Received MsgId -- ID = 0x%04X"

#define CF_CMD_LEN_ERR_EID 4
CFE_EVS_ERROR, "Cmd Msg with Bad length Rcvd: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d"

#define CF_NOOP_CMD_EID 5
CFE_EVS_INFORMATION, "CF No-op command, Version %d.%d.%d.%d"

#define CF_RESET_CMD_EID 6
CFE_EVS_DEBUG, "Reset Counters command received - Value %u"

#define CF_FILE_IO_ERR1_EID 7
CFE_EVS_ERROR, "Unable to open file = %s!"

#define CF_CR_PIPE_ERR_EID 8
CFE_EVS_ERROR, "Error Creating SB Pipe,RC=0x%08X"

#define CF_SUB_REQ_ERR_EID 9
CFE_EVS_ERROR, "Error Subscribing to HK Request,RC=0x%08X"

#define CF_SUB_CMD_ERR_EID 10
CFE_EVS_ERROR, "Error Subscribing to CF Gnd Cmds,RC=0x%08X"

#define CF_RCV_MSG_ERR_EID 11
CFE_EVS_ERROR, "CF_APP Exiting due to CFE_SB_RcvMsg error 0x%08X"

#define CF_FILE_IO_ERR2_EID 12
CFE_EVS_ERROR, "Unable to create file = %s!"

#define CF_REMOVE_ERR1_EID 13
CFE_EVS_ERROR, "Could not remove filename = %s." (in rename attempt)

#define CF_LOGIC_NAME_ERR_EID 14
CFE_EVS_ERROR,
"The filename %s size could not be retrieved because it does not exist." (in Filesize callback)

#define CF_CFDP_ENGINE_DEB_EID 15
CFE_EVS_DEBUG, All engine Debug events

#define CF_CFDP_ENGINE_INFO_EID 16
CFE_EVS_DEBUG, All engine Info events

#define CF_CFDP_ENGINE_WARN_EID 17
CFE_EVS_INFORMATION, All engine warning events

#define CF_CFDP_ENGINE_ERR_EID 18
CFE_EVS_ERROR, All engine error events

#define CF_FILE_IO_ERR3_EID 19
CFE_EVS_ERROR,
"File write error! Should have written = %d bytes but only wrote %d bytes to the file!"

#define CF_IN_TRANS_OK_EID 20
CFE_EVS_INFORMATION, "Incoming trans success %d.%d_%d,dest %s"
Registered for filtering, filter algorithm=unfiltered

#define CF_OUT_TRANS_OK_EID 21
CFE_EVS_INFORMATION, "Outgoing trans success %d.%d_%d,src %s"
Registered for filtering, filter algorithm=unfiltered

#define CF_IN_TRANS_FAILED_EID 22
CFE_EVS_ERROR, "Incoming trans %d.%d_%d %s,CondCode %s,dest %s"

#define CF_OUT_TRANS_FAILED_EID 23
CFE_EVS_ERROR, "Outgoing trans %d.%d_%d %s,CondCode %s,Src %s,Ch %d "

#define CF_MV_UP_NODE_EID 24
CFE_EVS_ERROR, "TransId %s_%d not found in CF_MoveUpNodeActiveToHistory"

#define CF_ENDIS_AUTO_SUS_CMD_EID 25
CFE_EVS_DEBUG, "Auto Suspend enable flag set to %u"

#define CF_IND_XACT_SUS_EID 26
CFE_EVS_INFORMATION, "Transaction Susupended %d.%d_%d,%s"
Registered for filtering, filter algorithm=unfiltered

#define CF_IND_XACT_RES_EID 27
CFE_EVS_INFORMATION, "Transaction Resumed %d.%d_%d,%s"
Registered for filtering, filter algorithm=unfiltered

#define CF_IND_XACT_FAU_EID 28
CFE_EVS_DEBUG, "Fault %d,%d.%d_%d,%s"

#define CF_IND_XACT_ABA_EID 29
CFE_EVS_INFORMATION, "Indication:Transaction Abandon %d.%d_%d,%s"

#define CF_KICKSTART_CMD_EID 30
CFE_EVS_DEBUG, "Kickstart cmd received, chan %u"

#define CF_REMOVE_ERR2_EID 31
CFE_EVS_ERROR, "Could not remove filename = %s." (in remove attempt)

#define CF_IND_ACK_TIM_EXP_EID 32

CFE_EVS_INFORMATION, "Flight Ack Timer Expired %d.%d_%d,%s"

#define CF_IND_INA_TIM_EXP_EID 33
CFE_EVS_INFORMATION, "Flight Inactivity Timer Expired %d.%d_%d,%s"

#define CF_IND_NACK_TIM_EXP_EID 34
CFE_EVS_INFORMATION, "Flight Nack Timer Expired %d.%d_%d,%s"

#define CF_IND_UNEXP_TYPE_EID 35
CFE_EVS_INFORMATION, "Unexpected Indication Type %d"

#define CF_FILE_CLOSE_ERR_EID 36
CFE_EVS_ERROR,
"Could not close file from callback function! OS_close Val = %d" (in Fclose callback)

#define CF_MACH_ALLOC_ERR_EID 37
CFE_EVS_ERROR, "CF:MachAlloc:AllocateQueueEntry returned NULL!\n"

#define CF_PLAYBACK_FILE_EID 38
CFE_EVS_DEBUG, "Playback File Cmd Rcvd,Cl %d,Ch %d,Pri %d,Pre %d,Peer %s,File %s"

#define CF_PUT_REQ_ERR1_EID 39
CFE_EVS_ERROR, "Put request for %s > MAX_REQUEST_STRING_LENGTH %u"

#define CF_PUT_REQ_ERR2_EID 40
CFE_EVS_ERROR, "Engine put request returned error for %s"

#define CF_CFGTBL_REG_ERR_EID 41
CFE_EVS_ERROR, "Error Registering Config Table,RC=0x%08X"

#define CF_CFGTBL_LD_ERR_EID 42
CFE_EVS_ERROR, "Error Loading Config Table,RC=0x%08X"

#define CF_CFGTBL_MNG_ERR_EID 43
CFE_EVS_ERROR, "Error from TBL Manage call for Config Table,RC=0x%08X"

#define CF_CFGTBL_GADR_ERR_EID 44
CFE_EVS_ERROR, "Error Getting Adr for Config Tbl,RC=0x%08X"

CF_TRANS_SUSPEND_OVRFLW_EID 45
CFE_EVS_ERROR, "Out Trans %u not suspended.Buffer overflow, max %u"

#define CF_FREEZE_CMD_EID 46
CFE_EVS_INFORMATION, "Freeze command received."
Registered for filtering, filter algorithm=unfiltered

#define CF_THAW_CMD_EID 47
CFE_EVS_INFORMATION, "Thaw command received."
Registered for filtering, filter algorithm=unfiltered

#define CF_CARS_CMD_EID 48
CFE_EVS_INFORMATION "%s command received.%s"

```
#define CF_CARS_ERR1_EID                49
CFE_EVS_ERROR, "%s Cmd Error,File %s Not Active"

#define CF_SET_MIB_CMD_EID              50
CFE_EVS_INFORMATION, "Set MIB command received.Param %s Value %s"

#define CF_GET_MIB_CMD_EID              51
CFE_EVS_INFORMATION, "Get MIB command received.Param %s Value %s"

#define CF_SUB_PDUS_ERR_EID             52
CFE_EVS_ERROR, "Error Subscribing to Incoming PDUs,RC=0x%08X"

#define CF_SND_Q_INFO_EID               53
CFE_EVS_DEBUG, "%s written:Size=%d,Entries=%d"

#define CF_FILEWRITE_ERR_EID            54
CFE_EVS_ERROR, "File write,byte cnt err,file %s,request=%d,actual=%d"

#define CF_SUB_WAKE_ERR_EID             55
CFE_EVS_ERROR, "Error Subscribing to Wakeup Cmd,RC=0x%08X"

#define CF_KICKSTART_ERR1_EID           56
CFE_EVS_ERROR, "Invalid Chan Param %u in KickstartCmd,Max %u"

#define CF_WR_CMD_ERR1_EID              57
CFE_EVS_ERROR, "Invalid Queue Param %u in WriteQueueInfoCmd,,1=active,2=history"

#define CF_WR_CMD_ERR2_EID              58
CFE_EVS_ERROR, "CF:Write Queue Info Error, Queue Value %u > max %u"

#define CF_WR_CMD_ERR3_EID              59
CFE_EVS_ERROR, "CF:Write Queue Info Error, Type Num %u not valid."

#define CF_SND_QUE_ERR1_EID             60
CFE_EVS_ERROR, "WriteQueueCmd:Error creating file %s, stat=0x%x"

#define CF_SND_TRANS_ERR_EID            61
CFE_EVS_ERROR, "Send Trans Cmd Error, %s not found."

#define CF_DEQ_NODE_ERR1_EID            62
CFE_EVS_ERROR, "Dequeue Node Cmd Error, %s not found."

#define CF_DEQ_NODE_ERR2_EID            63
CFE_EVS_CRITICAL, "DequeueNodeCmd:Trans %s is ACTIVE! Must send cmd again to remove"

#define CF_DEQ_NODE_ERR3_EID            64
CFE_EVS_CRITICAL, "DequeueNodeCmd:Trans %s is ACTIVE! Must send cmd again to remove"

#define CF_DEQ_NODE_ERR4_EID            65
CFE_EVS_ERROR, "Unexpected NodeType %d in Queue node"
```

#define CF_WR_CMD_ERR4_EID 66
CFE_EVS_ERROR,"CF:Write Queue Info Error, Channel Value %u > max %u"

Event ID 67 Unused

#define CF_DEQ_NODE1_EID 68
CFE_EVS_DEBUG, "DequeueNodeCmd %s Removed from %s Queue,Stat %d"

#define CF_DEQ_NODE2_EID 69
CFE_EVS_DEBUG, "DequeueNodeCmd %s Removed from Chan %u,%s Queue,Stat %d"

#define CF_PDU_RCV_ERR1_EID 70
CFE_EVS_ERROR, "PDU Rcv Error:PDU Hdr illegal size - %d, must be %d bytes"

#define CF_PDU_RCV_ERR2_EID 71
CFE_EVS_ERROR, "PDU Rcv Error:length %d exceeds CF_INCOMING_PDU_BUF_SIZE %d"

#define CF_PDU_RCV_ERR3_EID 72
CFE_EVS_ERROR, "cfdp_give_pdu returned error in CF_SendPDUToEngine"

#define CF_HANDSHAKE_ERR1_EID 73
CFE_EVS_ERROR, "SemGetId Err:Chan %d downlink PDUs cannot be throttled.0x%08X"

#define CF_SND_TRANS_CMD_EID 74
CFE_EVS_DEBUG, "CF:Sending Transaction Pkt %s"

#define CF_IND_FAU_UNEX_EID 75
CFE_EVS_ERROR, "Unexpected Condition Code %u in Trans Finished Indication"

#define CF_PB_FILE_ERR1_EID 76
CFE_EVS_ERROR, "Playback File Cmd Parameter error, class %d, chan %d"

#define CF_PB_FILE_ERR2_EID 77
CFE_EVS_ERROR, "CF:Playback File Cmd Parameter Error, Chan %u is not in use."

#define CF_PB_FILE_ERR3_EID 78
CFE_EVS_ERROR, "CF:Playback File Cmd Error, Chan %u Pending Queue is full %u."

#define CF_PB_FILE_ERR4_EID 79
CFE_EVS_ERROR, "CF:Playback File Cmd Error, File is Open:%s"

#define CF_PB_FILE_ERR5_EID 80
CFE_EVS_ERROR, "CF:Playback File Cmd Error, File is Already Pending or Active:%s"

#define CF_PB_FILE_ERR6_EID 81
CFE_EVS_ERROR, "CF:PB File Cmd Err, PeerEntityId %s must be 2 byte,dotted decimal fmt.ex 0.24"

#define CF_QDIR_INV_NAME1_EID 82
CFE_EVS_ERROR, "File not queued from %s,Filename not terminated or too long"

#define CF_QDIR_INV_NAME2_EID 83
CFE_EVS_ERROR, "File not queued from %s,sum of Pathname,Filename too long"

```
#define CF_MEM_ALLOC_ERR_EID            84
CFE_EVS_ERROR, "Memory Allocation Error, GetPoolBuf Returned 0x%x, dec %d"

#define CF_MEM_DEALLOC_ERR_EID          85
CFE_EVS_ERROR, "Deallocation failed for queue entry. Stat = %d"

#define CF_ENA_DQ_CMD_EID               86
CFE_EVS_DEBUG, "Channel %d Dequeue Enabled"

#define CF_DQ_CMD_ERR1_EID              87
CFE_EVS_ERROR, "Enable Dequeue Cmd Param Err Chan %d,Max is %d"

#define CF_DIS_DQ_CMD_EID               88
CFE_EVS_DEBUG, "Channel %d Dequeue Disabled"

#define CF_DQ_CMD_ERR2_EID              89
CFE_EVS_ERROR, "Disable Dequeue Cmd Param Err Chan %d,Max is %d"

#define CF_ENA_POLL_CMD1_EID            90
CFE_EVS_DEBUG, "All In-use Polling Directories on Channel %d Enabled"

#define CF_ENA_POLL_CMD2_EID            91
CFE_EVS_DEBUG, "Polling Directory %d on Channel %d Enabled"

#define CF_ENA_POLL_ERR1_EID            92
CFE_EVS_ERROR, "Channel Param Err in EnPollCmd.Chan %d,Max %d"

#define CF_ENA_POLL_ERR2_EID            93
CFE_EVS_ERROR, "Directory Param Err in EnPollCmd.Dir %d,0-%d and 255 allowed"

#define CF_DIS_POLL_CMD1_EID            94
CFE_EVS_DEBUG, "All In-use Polling Directories on Channel %d Disabled"

#define CF_DIS_POLL_CMD2_EID            95
CFE_EVS_DEBUG, "Polling Directory %d on Channel %d Disabled"

#define CF_DIS_POLL_ERR1_EID            96
CFE_EVS_ERROR,
"Channel Param Err in DisPollCmd.Chan %d,Max %d"
"Directory Param Err in DisPollCmd.Dir %d,0-%d and 255 allowed"

#define CF_DIS_POLL_ERR2_EID            97
CFE_EVS_ERROR, "Directory Param Err in DisPollCmd.Dir %d,0-%d and 255 allowed"

#define CF_OPEN_DIR_ERR_EID             98
CFE_EVS_ERROR, "Playback Dir Error %d,cannot open directory %s"

#define CF_QDIR_NOMEM_EID               99
CFE_EVS_ERROR,
"PB File %s Cmd Ignored,Error Allocating Queue Node."
"PB Dir %s Aborted,Error Allocating Queue Node."
```

Event ID 100 Unused

#define CF_QDIR_PQFUL_EID 101
CFE_EVS_ERROR,"Queue Dir %s Aborted,Ch %d Pending Queue is Full,%u Entries"

#define CF_IN_TRANS_START_EID 102
CFE_EVS_INFORMATION, "Incoming trans started %d.%d_%d,dest %s"
Registered for filtering, filter algorithm=unfiltered

#define CF_OUT_TRANS_START_EID 103
CFE_EVS_INFORMATION, "Outgoing trans started %d.%d_%d,src %s"
Registered for filtering, filter algorithm=unfiltered

#define CF_SET_MIB_CMD_ERR1_EID 104
CFE_EVS_ERROR,
"Cannot set OUTGOING_FILE_CHUNK_SIZE(%d) > CF_MAX_OUTGOING_CHUNK_SIZE(%d)"

#define CF_SET_MIB_CMD_ERR2_EID 105
CFE_EVS_ERROR, "Cannot set Flight Entity Id to %s, must be 2 byte, dotted decimal fmt"

#define CF_TBL_VAL_ERR1_EID 106
CFE_EVS_ERROR, "Cannot set FlightEntityId to %s, must be 2 byte, dotted decimal fmt like 0.24"

#define CF_TBL_VAL_ERR2_EID 107
CFE_EVS_ERROR,
"Cannot set IncomingPDUMsgId 0x%X > CFE_SB_HIGHEST_VALID_MSGID 0x%X"

#define CF_TBL_VAL_ERR3_EID 108
CFE_EVS_ERROR,
"Cannot set OUTGOING_FILE_CHUNK_SIZE(%d) > CF_MAX_OUTGOING_CHUNK_SIZE (%d)"

#define CF_TBL_VAL_ERR4_EID 109
CFE_EVS_ERROR, "Ch%d EntryInUse %d must be 0-unused or 1-in use"

#define CF_TBL_VAL_ERR5_EID 110
CFE_EVS_ERROR, "Ch%d DequeueEnable %d must be 0-disabled or 1-enabled"

#define CF_TBL_VAL_ERR6_EID 111
CFE_EVS_ERROR,
"Cannot set Ch%d Poll %d PeerEntityId to %s,must be 2 byte, dotted decimal fmt like 0.24"

#define CF_TBL_VAL_ERR7_EID 112
CFE_EVS_ERROR,
"Cannot set Ch%d OutgoingPduMsgId 0x%X > CFE_SB_HIGHEST_VALID_MSGID 0x%X"

#define CF_TBL_VAL_ERR8_EID 113
CFE_EVS_ERROR, "Ch%d,PollDir%d EntryInUse %d must be 0-unused or 1-in use"

#define CF_TBL_VAL_ERR9_EID 114
CFE_EVS_ERROR, "Ch%d,PollDir%d EnableState %d must be 0-disabled or 1-enabled"

```
#define CF_TBL_VAL_ERR10_EID            115
CFE_EVS_ERROR, "Chan %d,PollDir%d Class %d must be 1-unreliable or 2-reliable"

#define CF_TBL_VAL_ERR11_EID            116
CFE_EVS_ERROR, "Ch%d,PollDir%d Preserve %d must be 0-delete or 1-preserve"

#define CF_TBL_VAL_ERR12_EID            117
CFE_EVS_ERROR, "Ch%d,PollSrcPath must be terminated,have no spaces,slash at end"

#define CF_TBL_VAL_ERR13_EID            118
CFE_EVS_ERROR, "Ch%d,PollDstPath must be terminated,have no spaces,slash at end"

#define CF_TBL_VAL_ERR14_EID            119
CFE_EVS_ERROR, "Total Validation Errors - %d for CF Configuration Table"

#define CF_NO_TERM_ERR_EID              120
CFE_EVS_ERROR, "Unterminated string found in %s"

#define CF_SET_POLL_PARAM_ERR1_EID  121
CFE_EVS_ERROR, "Invalid Chan Param %u in SetPollParamCmd,Max %u"

#define CF_SET_POLL_PARAM_ERR2_EID  122
CFE_EVS_ERROR, "Invalid PollDir Param %u in SetPollParamCmd,Max %u"

#define CF_SET_POLL_PARAM_ERR3_EID  123
CFE_EVS_ERROR, "Invalid Class Param %u in SetPollParamCmd,Min %u,Max %u"

#define CF_SET_POLL_PARAM_ERR4_EID  124
CFE_EVS_ERROR, "Invalid Preserve Param %u in SetPollParamCmd,Max %u"

#define CF_SET_POLL_PARAM_ERR5_EID  125
CFE_EVS_ERROR, "SrcPath in SetPollParam Cmd must be terminated,have no spaces,slash at end"

#define CF_SET_POLL_PARAM_ERR6_EID  126
CFE_EVS_ERROR, "DstPath in SetPollParam Cmd must be terminated,have no spaces,slash at end"

#define CF_SET_POLL_PARAM_ERR7_EID  127
CFE_EVS_ERROR, "PeerEntityId %s in SetPollParam Cmd must be 2 byte,dotted decimal fmt.ex 0.24"

#define CF_SET_POLL_PARAM1_EID          128
CFE_EVS_DEBUG, "SetPollParam Cmd Rcvd,Ch=%u,Dir=%u,Cl=%u,Pri=%u,Pre=%u"

#define CF_SND_CFG_CMD_EID              129
CFE_EVS_DEBUG, "CF:Sending Configuration Pkt"

#define CF_QD_ERR1_EID                  130
CFE_EVS_ERROR, "QueueDirFiles Error, Invalid Class %d"

#define CF_QD_ERR2_EID                  131
CFE_EVS_ERROR, "QueueDirFiles Error, Invalid Chan %d"

#define CF_QD_ERR3_EID                  132
```

CFE_EVS_ERROR, "QueueDirFiles Error, Invalid CmdOrPoll Param %d"

Event ID 133 Unused

#define CF_PLAYBACK_DIR_EID 134
CFE_EVS_DEBUG, "Playback Dir Cmd Rcvd,Ch %d,Cl %d,Pri %d,Pre %d,Peer %s, Src %s,Dst %s"

#define CF_PB_DIR_ERR1_EID 135
CFE_EVS_ERROR, "Playback Dir Cmd Parameter error,class %d,chan %d,preserve %d"

#define CF_PB_DIR_ERR2_EID 136
CFE_EVS_ERROR, "CF:Playback Dir Cmd Parameter Error, Chan %u is not in use."

#define CF_PB_DIR_ERR3_EID 137
CFE_EVS_ERROR, "SrcPath in PB Dir Cmd must be terminated,have no spaces,slash at end"

#define CF_PB_DIR_ERR4_EID 138
CFE_EVS_ERROR, "DstPath in PB Dir Cmd must be terminated,have no spaces,slash at end"

#define CF_PB_DIR_ERR5_EID 139
CFE_EVS_ERROR, "CF:PB Dir Cmd Err,PeerEntityId %s must be 2 byte,dotted decimal fmt.ex 0.24"

#define CF_PURGEQ_ERR1_EID 140
CFE_EVS_ERROR, "PurgeQueueCmd Err:Cannot purge Incoming ACTIVE Queue"

#define CF_PURGEQ_ERR2_EID 141
CFE_EVS_ERROR, "Invalid Queue Param %u in PurgeQueueCmd"

#define CF_PURGEQ_ERR3_EID 142
CFE_EVS_ERROR, "PurgeQueueCmd Err:Cannot purge Outgoing ACTIVE Queue"

#define CF_PURGEQ_ERR4_EID 143
CFE_EVS_ERROR, "Invalid Queue Param %u in PurgeQueueCmd,Max %u"

#define CF_PURGEQ_ERR5_EID 144
CFE_EVS_ERROR, "Invalid Chan Param %u in PurgeQueueCmd,Max %u"

#define CF_PURGEQ_ERR6_EID 145
CFE_EVS_ERROR, "Invalid Type Param %u in PurgeQueueCmd,must be uplink %u or playback %u"

#define CF_PURGEQ1_EID 146
CFE_EVS_INFORMATION, "PurgeQueueCmd Removed %u Nodes from Uplink History Queue"

#define CF_PURGEQ2_EID 147
CFE_EVS_INFORMATION, "PurgeQueueCmd Removed %u Nodes from Chan %u,%s Queue"

#define CF_WRACT_ERR1_EID 148
CFE_EVS_ERROR, "CF:Write Active Cmd Error,Type Value %d > max %d"

#define CF_WRACT_ERR2_EID 149
CFE_EVS_ERROR, "SendActiveTransCmd:Error creating file %s, stat=0x%x"

#define CF_WRACT_TRANS_EID 150
CFE_EVS_DEBUG, "%s written:Size=%d,Entries=%d"

#define CF_TBL_LD_ATTEMPT_EID 151
CFE_EVS_ERROR, "CF Config Tbl cannot be updated! Load attempt must be aborted!"

#define CF_OUT_SND_ERR1_EID 152
CFE_EVS_ERROR, "Dropping PDU,cannot find channel number for TransId %d.%d_%d"

#define CF_OUT_SND_ERR2_EID 153
CFE_EVS_ERROR, "Dropping PDU,Ch %d,Msg Size %d > Max 65535,TransId %d.%d_%d"

#define CF_OUT_SND_ERR3_EID 154
CFE_EVS_ERROR, "Dropping PDU,Ch %d,Failed to get SB buffer %d,TransId %d.%d_%d"

#define CF_QDIR_ACTIVEFILE_EID 155
CFE_EVS_DEBUG, "File %s not queued because it's active or pending"

#define CF_QDIR_OPENFILE_EID 156
CFE_EVS_INFORMATION, "File %s not queued because it's open"

#define CF_INV_FILENAME_EID 157
CFE_EVS_ERROR, "Filename in %s must be terminated and have no spaces"

Event ID 158 Unused

#define CF_QUICK_ERR1_EID 159
CFE_EVS_ERROR, "Quick Status Cmd Error,Trans %s Not Found"

#define CF_QUICK_CMD_EID 160
CFE_EVS_INFORMATION "Trans %s_%u %s Stat=%s,CondCode=%s"

Appendix E - Constraints

1. All Entity ID's must be formatted as two-byte, dotted-decimal notation. Asist, ITOS and CF use a similar engine designed by GSFC Code 583. One factor that dictates the size of the outgoing PDU Header size, is the Entity ID format. It is necessary to set the CF application entity ID to the format specified above so that the PDU header size of outgoing PDUs is 12 bytes. Entity ID formats other than what is specified will likely produce compiler packing that may shift bytes in the outgoing packets. An example of a two-byte, dotted-decimal format is 2.25 or 0.24. The limits are 0.0 to 255.255.

2. The CFDP standard CCSDS 727.0-B-4 allows a variety of data-type sizes for Source and Destination Entity ID's, Transaction ID and PDU Header length. Currently, this application does not support the following:
    a. an Entity ID length other than 2 bytes
    b. a transaction ID length other than 4 bytes
    c. a PDU header size other than 12 bytes.

3. The stack size for the CF application must be monitored and must be no less than 16384 bytes. Depending on the CF configuration, the stack size may need to be set higher than 16384. The stack size is specified in one of two places.
    a.The cfe_es_startup.scr file that is located in the /mission/build/xxx/exe area.
    b. The CFE ES command CFE_ES_START_APP_CC which is used to start the application.

4. Poll directories must not have subdirectories, otherwise errors will occur at put request time.

5. All files in polling directories must be closed.

6. The same file cannot be sent on two channels at same time.

7. Spaces are not allowed in filenames.

8. Segmentation control as described in the blue book is not supported by this version of CFS CF application. All outgoing transactions will have the segmentation control bit in the meta-data PDU set to - 'Recorded Boundaries Not Respected'.

9. This version of CF does not support keep alive procedures that are detailed in section 4.1.6.5 of CFDP CCSDS 727.0-B-4 Blue Book.

10. Invalid file structures are not supported by this version of the CF application. Invalid file structures are described in 4.1.6.1.1.8 of CFDP CCSDS 727.0-B-4 Blue Book and apply only when record boundaries are respected (see segmentation control above).

11. The CF application will not issue an invalid transmission mode fault. Both transmission modes, unacknowledged (class 1) and acknowledged (class 2) are supported by the CF application.

12. The CFDP Application will fail on startup if the following conditions are not met:

Unable to create a Software Bus Pipe
Unable to subscribe to the CF Command Message
Unable to subscribe to the CF Housekeeping Request Message
Unable to register for cFE Event Services
Unable to register the CF Configuration Table with cFE Table Services
Unable to load the CF Configuration Table with a defaut table file
Unable to acquire a pointer to the CF Configuration Table

Each one of these conditions will generate a unique event message and will cause the CF application to terminate before processing any CF command pipe messages.

Appendix F - Troubleshooting

1.Problem - Feds not passing all PDUs to Asist CFDP Engine
Solution - For GPM FSW 3.0+, FEDs needs to pass 3 new MsgIds to Asist CFDP Engine.
How To - Check that the FEDS Config file
/mission/gpm/config_data/exp_01_dist_file has the following
defined for VC0,VC1, VC2 and VC3.

output_stream=HK
vcid=0, apid=(0x0b4-0x0b7)
vcid=1, apid=(0x0b4-0x0b7)
vcid=2, apid=(0x0b4-0x0b7)
vcid=3, apid=(0x0b4-0x0b7)

If feds config file is changed, recompile by typing 'dhds 01' at the cmd prompt.

2. Problem - Ground engine not responding (no white messages in event window during a downlink transfer, flight Ack-Timeouts occur)
Solution - Add VC1,VC2 and VC3 MsgIds to PDU list
How To – Add MsgIds to $WORK/prc/gpm_setup_cfdp.prc

3. Problem - Inactivity Timeouts reported by ground engine
If this is occurring when flight CF has Auto-suspend enabled (see hk tlm page), it may be due to a bug found in the engine. When engine is frozen, all timers are paused. But if a meta-data pdu is received when the engine is frozen, the inactivity timer for this new transaction continues to count down.
Solution - Set inactivity timeout so high, that it will never occur.
How To - From STOL prompt, cfdp_dir "inactivity_timeout=5400"

4. Problem - Flight side, Ack Timeouts reported by CF
Check that the Flight Ack Timeout is set properly (30 seconds). Also check that the following ground engine parameters are set properly. In $WORK/db/cfdp.machinename.config, the milliseconds-between-engine-cycles should be set to 50. The milliseconds-between-green-lights should be set to 45.
Solution – Set Ack Timeout to be 30 seconds or adjust the millisecond delay values as mentioned above.
How To - See description above.

5. Problem - Ground engine not responding (no white messages in event window during a downlink transfer, flight Ack-Timeouts occur)

Solution - Execute the setup procedure to initialize the ground CFDP engine.

How To – run proc 'gpm_setup_cfdp'

6. Problem - Ground engine not responding (no white messages in event window during a downlink transfer, flight Ack-Timeouts occur)

Solution - Every Asist machine needs a cfdp.hostname.config file located in the $WORK/db directory. In this file, the following parameter needs to contain the hostname:

dest_of_outgoing_pdus=hostname:CFDP2CPKT

7. Problem - CF displaying error event CF_PDU_RCV_ERR1_EID, "PDU Rcv Error:PDU Hdr illegal size - %d, must be %d bytes".

Solution - Change format of Ground Engine, Source Entity ID to the required two-byte dotted decimal format. See Entity ID definition for more information. This is an indication that the peer engine, entity ID is not set to the required two-byte dotted decimal format. The entity ID format has a direct affect on the size of the PDU header.

How To - If the peer engine is located in Asist, check the entity ID defined in the $WORK/db/cfdp.hostname.config file. For example, if the entity ID is set to 23, change it to 0.23

8. Problem - Ground engine not responding (no white messages in event window during a downlink transfer, flight Ack-Timeouts occur)

Solution: On all Asist machines, check that the following config parameter is set to 'no' in the $WORK/db/cfdp.config file as well as the $WORK/db/cfdp.hostname.config file.

tlm_use_local_response = no

Appendix G - Known Problems

1. The engine version (3.1.a.1) used on CF and on the Asist ground station has an anomaly when the engine is in the 'freeze' state. When the engine is in the state of 'freeze', the timers for all active transactions are paused, no PDUs will be sent out, but incoming PDUs will be accepted. The problem is that the inactivity timer for an incoming transaction that begins after the engine is frozen is not paused as it should be, so it may result in an inactivity timeout. A change request has been submitted to the Asist team.

2. Suspend Request Counter (SCX_CPU1_CF_SuspendNum) in CF housekeeping telemetry does not reflect the number of transactions suspended. This mnemonic has been removed from the page. A CFS

FSW DCR has been entered. There is a similar telemetry point (SCX_CPU1_CF_NumSuspended) which comes directly from the engine that does keep an accurate count of suspended transactions.

3. CF has a compatibility issue with the cFE that results in a compiler error. CF version 2.1 is not compatible with cFE 6.1.1. This problem was fixed in CF version 2.2.x. This also means that CF 2.2.x is not compatible with 6.0.x and earlier. If compatibility becomes an issue and it is not possible to get the correct version, the error can be resolved by changing the CF code. Add a sixth parameter to the CFE_ES_PoolCreateEx call. Make the value of the new parameter, one (1). This tells CF to use the semaphore provided by the memory pool.

4. The semaphore value on the housekeeping page does not display the correct value. A FSW DCR has been entered. It will likely be fixed in the next OSAL release.