NASA GSFC FLIGHT SOFTWARE SYSTEMS BRANCH

FSW VERSION DESCRIPTION DOCUMENT

CFS CFDP (CF) APPLICATION

BUILD: CF 3.0.0

RELEASE DATE: 9/28/2021

1.0 FSW VERSION DESCRIPTION

1.1 PURPOSE AND SUMMARY

This build is a major build of the CFDP (CF) application that streamlines the implementation of CFDP for flight environments and provides compatibility with cFS Caelum. This version of CFDP is a major overhaul of the CF code, and does include **breaking changes.**

1.2 NEW/CHANGED FUNCTIONALITY IN THIS VERSION

CF v3.0.0 provides the same high-level, CFDP file transfer functionality as previous versions of the CF application, but there are several notable changes from the previous version. These changes are summarized below, and additional detail is available in the supplemental documentation provided with this release.

Key changes from CF v2.x.x

- CF 3.0.0 does not have text-based ground commands using string arguments.
- CF 3.0.0 does not use a cFS memory pool. The app is configured with limits both in platform config for static memory limits and the configuration table for dynamic timing and functionality limits.
- CF 3.0.0 is a light-weight flight-only app. It does not provide any ground engine support, and therefore has a much small code footprint.

1.3 MISSING PLANNED FEATURES AND KNOWN PROBLEMS

Table 1.3-1 identifies currently open DCRs against CF v3.0.0 that are not addressed in this release. DCRs against prior releases of CF are not included in this table, but are available by request to the cFS Program Team, cfs-program@nasa.onmicrosoft.com.

Table 1.3-1 – Currently open DCRs

Key	Summary	Description
GSFCCFS-1190	Cleanup CF v3.0 Perf ID handling	Encompasses several findings from the CF v3.0 code review: - Remove unused perf id CF_DIRREAD_PERF_ID - Determine whether perf ids CF_PDU_RCVD_PERF_ID and CF_VCxPDUSENT_PERF_ID should be added (currently used in PACE) - Ensure that CFE_ES_PerfLogExit is called when the app exits

GSFCCFS-1194	Improve commenting throughout CF v3.0	CF v3.0 needs improved commenting throughout. Specifically: - file comments in every file describing the purpose of the file - explanations and limits for all configuration parameters - ensure that all comments are accurate
GSFCCFS-1196	Investigate whether TX and RX PDU sizes need to be different	Need to investigate uses cases where TX and RX PDU sizes need to be different. Need to investigate the impact of making a change to allow that.
GSFCCFS-1209	CF Doxygen Documentation Needs Updates	The CF doxygen documentation is inaccurate for v3.0.
GSFCCFS-1217	CF-3.0 having a problem building unit tests with unlikely((x))	When building with `make ENABLE_UNIT_TESTS=TRUE SIMULATION=native` and `NDEBUG` defined I get a `undefined reference unlikely` error. Determine if we need to keep `unlikely` and if so, what needs done to get it to build with the unit tests.

GSFCCFS-1236	Review use of 'unlikely'	In cf_assert.h the NDEBUG #ifdef runs in 'BUILDTYPE=release' mode which replaces CF_Assert with
		if(unlikely((x))) CF_HandleAssert(__FILE__, __LINE__);
		This line is incorrect, because the other path uses assert(x)
		However, assert does nothing when true, the unlikely path causes assert when true. This needs updated with (!x)
		
		Also, there are calls to CF_Assert using "short-circuit" Boolean statements, these do not "short-circuit" because of the unlikely.
		
		unlikely is most commonly defined as:
		__builtin_expect(!!(x), 0)

GSFCCFS-1689	CF: CF_CFDP_SendNak ph value can never be NULL	The CF_CFDP_SendNak function receives it's header from `&((pdu_s_msg_t*)CF_AppData.engine.out.msg)->ph` but even when CF_AppData.engine.out.msg == NULL, ph is 0x10. The check if ph is NULL (`if (!ph)`) will never be able to fire. The check should be changed to see if CF_AppData.engine.out.msg is NULL instead; this will ensure that if/when it is NULL a ph value of 0x10 will not be used. Note: This is probably not possible in the field, a "nack" should never be able to be returned without msg being populated, but in that case there is no reason to verify ph is not NULL.

GSFCCFS-1701	CF Purge command does not appear to be hooked in	CF comments make reference to a purge command, and functions exist for that command. However, the command code referenced in the comments (CF_PURGE_QUEUE_CC) does not appear to exist and there does not appear to be an equivalent one. https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_cf/-/blob/cf-3.0/fsw/src/cf_cmd.c#L1054 https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_cf/-/blob/cf-3.0/fsw/src/cf_cmd.c#L1081
GSFCCFS-1703	Unused event ID	The event ID CF_EID_ERR_PDU_BAD_RX_MSG_SIZE appears to be unused.
GSFCCFS-1719	CF method CF_CList_Remove appears to accept bad arguments	A bad node is passed to CF_CList_Remove, but it carries on unaware. This was found because branch 3 of `if((node->next==node)&&(node->prev==node))` can only be covered by a test that passes this situation, node->next == node, node->prev != node. The issue: should a single node enter, that has a node->next == node (meaning: pointing to itself), but a node->prev != node (meaning: NOT pointing to itself), the code continues as if this is a valid state. It may be impossible for a bad node to enter here, but that is not apparent at the unit test level for CF_CList_Remove nor is the Doxygen brief clear on that assumption.

GSFCCFS-1726	CF method CF_CFDP_MsgOutGet returns 0x10 when it should be NULL	CF_CFDP_MsgOutGet has a path where it will return a 0x10 if the CF_AppData.engine.out.msg is NULL. There are two if blocks after the msg NULL verification that if neither come back true the assignment on line 382 of cf_cfdp.c:
		
		`ret = &((pdu_s_msg_t*)CF_AppData.engine.out.msg)->ph;`
		
		will make ret == 0x10, not NULL.
		The description of the return statement shows that this is not correct:
		
		`\retstmt Pointer to a pdu_header_t within a software bus buffer on success. Otherwise NULL.`

GSFCCFS-1727	CF CF_CFDP_MsgOutGet method double checks msg for NULL	The function CF_CFDP_MsgOutGet in cf_cfdp.c checks `if(!CF_AppData.engine.out.msg)` (line 356) as a top if block; however, it then checks it again as part of the if block `if(!CF_AppData.engine.out.msg&&((CF_AppData.config_table->chan[t->chan_num].sem_name[0]&&(OS_CountSemTimedWait(c->sem_id, 0)==OS_SUCCESS))||(!CF_AppData.config_table->chan[t->chan_num].sem_name[0])))` (starting on line 365 <- yep not a typo line 356 and 365, funny), but there is nothing in between that could change it. It would appear that this check in the second `if` is not required.

GSFCCFS-1728	CF function CF_CFDP_IsSender(transaction_t *ti) is odd one out because it uses ti	Every other function in cfdp.c that uses a transaction_t* as an argument names it 't', but CF_CFDP_IsSender uses ti.
GSFCCFS-1732	CF function CF_CFDP_ProcessPlaybackDirectory has an oddly bracketed block	In cfdp.c the function CF_CFDP_ProcessPlaybackDirectory has a block of code with no if/while et. Just a bracketed block of code that does not seem to have a reason to be bracketed.
GSFCCFS-1733	CF method CF_Assert call branches cannot be tested during unit testing	Calls to CF_Assert() cannot be stubbed by unit testing. The reasons: 1. CF_Assert uses assert which will kill the running process and subsequently the unit test runner when called by code under test. 2. CF_Assert can be stubbed a. The stub cannot stop the return to the code under test. b. The value required to cover the branch will cause a segfault upon return -- as the intent of CF_Assert is to kill the app before it takes cFS with it. c. There is currently no allowable way to have these branches tested during automated runs. d. It is possible to run ad hoc tests to show the assert occurs, but the unit test will stop and not return to the original test code as the automated tests do.
GSFCCFS-1734	CF method CF_CFDP_ReceiveMessage will always segfault on line 1289	Line 1289: ++CF_AppData.hk.channel_hk[t->chan_num].counters.recv.dropped; The only path to get into this line runs through an if(t) failure, which means t will always be NULL at this point. If it is changed to just chan_num (a variable defined within the function) it will not fail. However, it is not known if that is the intent. There is a previous `if` block that redefines the variable t, from a transaction_t* to a transaction_t, but its scope is only within that `if` block. Thus when we reach line 1289, t is the transaction_t* type and will always be NULL.

2.0 DELIVERED PRODUCTS

Table 2-1 identifies the locations of FSW products relevant to this FSW Build. The version or date of the Build and where the product can be located are provided. Changes from a previous VDD are identified.

Table 2-1 – Delivered Products and their Locations

Software Element	Changed with this Version?	New Version or Date	Location
Source Code of this FSW Build	Yes	3.0.0	https://github.com/nasa/cf
Doxygen Documentation	Yes	N/A	https://github.com/nasa/cf
Unit Test Data	Yes	3.0.0	https://github.com/nasa/cf
FSW Make Files	Yes	3.0.0	https://github.com/nasa/cf

3.0 INSTALLATION PROCEDURES

In order to build and install the CS application, it must be added to the cFE CMake build system. This is done by modifying the TGTX_APPLIST in the cFE targets.cmake file. This is shown in the trivial example below.

```
SET(TGT1_NAME cpu1)
SET(TGT1_APPLIST cf)
SET(TGT1_FILELIST cfe_es_startup.scr)
```

After CF is added to the targets.cmake file, it is built and installed using the standard cFE CMake build instructions. These instructions are available in cFE CMake documentation:

https://github.com/nasa/cFE/blob/main/cmake/README.md

4.0 CONFIGURATION SUMMARY AND VERSION IDENTIFICATION

This software can be found in the CF GitHub repository (https://github.com/nasa/CF) under the tag “CF-3.0.0”.

Verification of the version can be done by sending a CF NOOP command that produces an event message containing the version information. In addition, the initialization event message generated during the application startup provides the version information.

ACRONYMS

ACS ..... Attitude Control System
C&DH.....Command and Data Handling
cFS……..Core Flight System
CM .....Configuration Management
CS……...Checksum
COTS..... Commercial Off-The-Shelf
CRC……Cyclic Redundancy Check
DCR ..... Discrepancy/Change Request
ETU..... Engineering Test Unit
FSB.....Flight Software Branch
FSW.....Flight Software
H&S ……..……Health & Safety
I&T ..... Integration & Test
PSP ……Platform Support Package
RTOS.....Real-Time Operating System
T&C..... Telemetry and Command
URL.....Universal Resource Locator
VDD ..... Version Description Document