NASA GSFC FLIGHT SOFTWARE SYSTEMS BRANCH

FSW VERSION DESCRIPTION DOCUMENT

CFS STORED COMMANDS (SC) APPLICATION

BUILD: SC 3.1.0

RELEASE DATE: SEPTEMBER 28, 2021

1.0 FSW VERSION DESCRIPTION

1.1 PURPOSE AND SUMMARY

The purpose of this build is to continue to refine the cFS Stored Commands (SC) application product. This build provides various bug fixes and enhancements but does not include any new functionality. The primary purpose of this release is to ensure compatibility between the SC application and cFS Caelum.

This document serves as the notification of the Build 3.1.0 release of the cFS SC application.

1.2 NEW/CHANGED FUNCTIONALITY IN THIS VERSION

Table 1.2-1 identifies the DCRs that have been implemented in this FSW version. For each DCR the "Key" column shows the corresponding DCR in the GSFC cFS tracking system.

Table 1.2-1 – DCRs Implemented in this Version

Key	Summary	Description
GSFCCFS-1185	SC has static code analysis findings	In analysis done on 7/10/2020, CodeSonar flagged the attached findings.
GSFCCFS-1346	SC RTS command length increments incorrectly	In sc_state.c, lines 193-194, there appears to be a typo: CFE_MSG_GetSize(&CmdPtr->Msg, &CmdLength); CmdLength = SC_RTS_HEADER_SIZE; Should be: CFE_MSG_GetSize(&CmdPtr->Msg, &CmdLength); CmdLength += SC_RTS_HEADER_SIZE; This may explain an issue a customer is experiencing in which they can only send the first command of an RTS. Appears that this bug was introduced after SC 3.0.0 was tagged while completing ticket GSFCCFS-1263
GSFCCFS-1419	Incorrect check for valid RTS number	RTSs are numbered 1 to SC_NUMBER_OF_RTS inclusive. If you command 0 you'll get a invalid RTS 65535. Internally they are 0 to (SC_NUMBER_OF_RTS-1) but this assumes 0-1 will result in bad RTS number. Shouldn't the check really be for 1 to SC_NUMBER_OF_RTS inclusive prior to minus 1? Probably true in other RTS command functions. ---Code Snippet--- void SC_StartRtsCmd (CFE_SB_MsgPtr_t CmdPacket) { uint16 RtsIndex; /* rts array index */ CFE_SB_MsgPtr_t RtsEntryCmd; /* pointer to an rts command

		*/
		SC_RtsEntryHeader_t *RtsEntryPtr;
		uint16 CmdLength; /* the length of the 1st cmd */
		
		/*
		** Verify command packet length...
		*/
		if (SC_VerifyCmdLength(CmdPacket, sizeof(SC_RtsCmd_t)))
		{
		/* convert RTS number to RTS array index */
		RtsIndex = ((SC_RtsCmd_t *)CmdPacket) -> RtsId - 1;
		/*
		** Check start RTS parameters
		*/
		if (RtsIndex < SC_NUMBER_OF_RTS)
		{
		---------- 	
		
		It looks like there is an underflow if the RTS is 0. The event sent by SC properly shows RTS 0 instead of RTS 65535 because it uses the command parameter, not the computed value.
		
		INFO dsp_seqprt 21-063-13:05:02.770: error SC 77: Start RTS 000 Rejected: Invalid RTS ID

GSFCCFS-1423	Migrate SC unit tests to distributed UT Assert	
GSFCCFS-1440	SC: Function arguments should use correct types and be "const" where possible	SC has quite a few places where function arguments can and should be "const". The entire application should be scrubbed, but specific instances pointed out in the code review are listed below: SC_StopAtsCmd SC_GroundSwitchCmd SC_JumpAtsCmd SC_ContinueAtsOnFailureCmd SC_AppendAtsCmd SC_ProcessCommand SC_TableManageCmd SC_ValidateAts SC_ParseRts SC_ValidateRts SC_VerifyAtsEntry SC_StartRtsCmd SC_StartRtsGrpCmd SC_StopRtsCmd SC_StopRtsGrpCmd SC_DisableRtsCmd SC_DisableRtsGrpCmd SC_EnableRtsCmd SC_EnableRtsGrpCmd

		In the SC_ValidateRts function, the parameter could be uint32* instead of a void* that is cast to a uint32*.
GSFCCFS-1441	Clarify comments in SC platform_inc header files	In sc_platform_cfg.h: - Clarify that SC_PACKET_MIN_SIZE and SC_PACKET_MAX_SIZE are specified in bytes, not words In sc_rts.h: - Add comment clarifying how RTS_ID_Spare3 - RTS_ID_Spare256 are intended to be used/customized by projects
GSFCCFS-1442	SC_PACKET_MAX_SIZE is not compatible with new alignment requirement	SC 3.0.0 enforces 32-bit alignment. The SC_PACKET_MAX_SIZE of 250 does not meet that alignment. The value should be updated and a check should be added to sc_verify.h to enforce the alignment.
GSFCCFS-1443	SC uses deprecated CFE_PSP_MemSet and CFE_PSP_MemCpy	Should replace CFE_PSP_MemSet with regular memset and CFE_PSP_MemCpy with regular memcpy.
GSFCCFS-1444	SC Formatting needs cleanup	There's irregular whitespace and indentation throughout the SC app that needs cleanup. Also, make sure that a consistent approach is taken to using "return" statements at the end of "void" functions.
GSFCCFS-1446	snprintf should be used instead of sprintf	In several places, sprintf is used in cases where the string being copied could exceed the size of the destination array. These instances should all use snprintf instead of sprintf.
GSFCCFS-1448	SC_LoadDefaultTables should log tables that fail to load	SC_LoadDefaultTables currently sends an event message that reports the number of RTSs successfully loaded. The event message should be modified to report the number of tables that failed to load. The function could also generate a debug event message for each load failure.
GSFCCFS-1449	Clarify comments in source file headers	In sc_app.h: -- typo "heats" on line 237 -- clarify difference between SC_AppData and SC_OperData In sc_loads.h: -- comment for SC_ParseRts is missing a return tag -- comment for SC_VerifyAtsEntry is missing an \endcode tag In sc_utils.h: -- the function for SC_GetAtsEntryTime is missing a comment entirely
GSFCCFS-1451	SC_RtsEntryHeader_t relies on removed OS_PACK macro	OS_PACK macro is removed as of OSAL Development Build: v5.1.0-rc1+dev184
GSFCCFS-1452	Toggling of ATSs needs to be clarified	There are two ATSs in SC: 0 and 1. There are several places in the code where the ATS is switched from one to the other. The switches need to be done using a consistent mechanism, and that

		mechanism should be made clear with a comment.
GSFCCFS-1454	Bounds checking is needed throughout SC	There are several places in the SC code where an RTS or ATS index is provided as a function parameter - proper bounds checking needs to be added.
GSFCCFS-1455	Add event message to SC_JumpAtsCmd	In the SC_JumpAtsCmd, add an event message for the else block covering the case where the jump time is less than or equal to the list entry. Need to make sure this case is not covered by the existing SC_JUMPATS_CMD_STOPPED_ERR_EID event that is sent after the else block.
GSFCCFS-1456	Comparisons should use values of the correct type	In SC_ContinueAtsOnFailureCommand, the State value is a uint16 but is being compared to "true" and "false". In SC_ProcessAtpCmd, TempAtsChar has a type of uint16, but is used as a char (including in the event message near the end of the function).
GSFCCFS-1457	Clarify comments throughout the source code	In sc_atsrq.c: -- In SC_StartAtsCmd (line 128) clarify comment to state that the SC_OperData.AtsCtrlBlckAddr->AtpState is updated in SC_KillAts. In sc_cmds.c: -- Clarify why handing of the SC_SWITCH_ATS_CC in SC_ProcessAtpCmd is different from the handling of all other commands. In sc_loads.c: -- Comment for SC_LoadRts is misleading In sc_state.c: -- Clarify why the loop in SC_GetNextRtsTime must go backwards (or at least why the behavior of going forward and breaking out is not equivalent)
GSFCCFS-1459	Errors in inline switch may not be completely handled	The following is an issue that needs investigation, and may or may not be an actual error. In SC_ProcessAtpCmd, a "Switch ATS" command is handled differently from all other commands. If a "Switch ATS" command is identified and the SC_InlineSwitch function fails, the following actions are taken to address the error: -- Set status to SC_FAILED_DISTRIB -- Increment ATS Cmd Error Counter -- Set LastAtsErrSeq -- Set LastAtsErrCmd In all other cases, a message is transmitted on the software bus. In the case of an error, the following actions are taken to address the error: -- Set status to SC_FAILED_DISTRIB

		-- Increment ATS Cmd Error Counter -- Set LastAtsErrSeq -- Set LastAtsErrCmd -- Send an event -- Abort the ATS The handling of these error cases is very similar, but not exactly the same. Investigation is needed to determine if an event should be sent and the ATS should be aborted if the Inline Switch fails.
GSFCCFS-1462	Use "else" clause to clarify code	In the SC_ProcessAtpCmd, the final call, to SC_GetNextAtsCommand(), is effectively a no-op if the ATS was aborted. It would make the code more readable if that call was wrapped in an "else" clause following the "if (AbortATS == true)" check.
GSFCCFS-1463	Track counters directly in the HK packet	SC currently maintains counters in the AppData struct and copies them into the housekeeping packet. SC is single-threaded, which makes it safe to maintain the counters directly in the housekeeping packet and avoid the extra copies.
GSFCCFS-1464	Replace hard-coded constants with macros	There are several instances of this throughout SC.
GSFCCFS-1465	Check for unreachable code	Need to check for unreachable code in SC_LoadAts. Readme file from old unit tests suggests that one of the else blocks may be unreachable.

GSFCCFS-1469	SC_StartRtsGrpCmd handles errors differently than StartRtsCmd	Both SC_StartRtsGrpCmd and SC_StartRtsCmd check the following condition for the RTS(s) they are starting:
		"if (SC_OperData.RtsInfoTblAddr[RtsIndex].DisabledFlag == false)
		{
		/* the requested RTS is not being used and is not empty */
		if (SC_OperData.RtsInfoTblAddr[RtsIndex].RtsStatus == SC_LOADED)"
		
		In the case of the SC_StartRtsCmd, if an individual RTS does not meet those conditions, the SC_AppData.RtsActiveErrCtr and SC_AppData.CmdErrCtr are both incremented. If the RTS does meet those conditions, the SC_AppData.CmdCtr is incremented (among other actions).
		
		However, in the case of the SC_StartRtsGrpCmd, if an individual RTS within the group does not meet those conditions, only the SC_AppData.CmdErrCtr is incremented - not the SC_AppData.RtsActiveErrCtr. Notably, the SC_AppData.CmdCtr is incremented in the success case for each RTS, further suggesting that the failure to increment RtsActiveErrCtr is in fact a defect.

GSFCCFS-1470	Investigate handling of states in SC_GetNextAtsCommand	SC_GetNextAtsCommand, the SC_EXECUTING and SC_STARTING states are handled differently, but it is unclear why.

GSFCCFS-1472	Use most up-to-date types for time	OSAL is moving to unified representation of time?
GSFCCFS-1473	Inconsistencies in sc_verify.h	Multiple inconsistencies in sc_verify.h: -"TRUE" and "FALSE" should be "true" and "false" respectively to be consistent with the rest of the code. - Several checks are separately checking of "!= TRUE" or "!= FALSE" when they could be using a combined condition for "!= TRUE && != FALSE" - The value of SC_NUMBER_OF_RTS should be checked against both the CFE_TBL_MAX_NUM_TABLES (currently done) and the upper limit of the uint16 type (not currently done) - Investigate whether a check could be added that could ensure the length of SC_RTS_FILE_NAME + 3(for %03d) + 4(for .tbl) + 1(null terminator) doesn't exceed OS_MAX_PATH_LEN?
GSFCCFS-1474	Format specifiers should work for larger MIDs	In the SC_ProcessCommand function (and potentially other places) the 0x04X format specifier is used for the MessageId field in an event message. This works for 16-bit message IDs, but not for the larger message IDs now allowable in cFE.
GSFCCFS-1483	SC does not build with eval-cert3	
GSFCCFS-1584	SC doxygen config file should be renamed for clarity	The filename "sc_config.txt" suggests that this a configuration file for the app itself as opposed to a configuration file for doxygen.
GSFCCFS-1637	SC doesn't work with extended message IDs	
GSFCCFS-1652	SC restart behavior is inconsistent	When Restarting the SC application (ES_RestartApp command), the first time, the ATP state is set to “Empty” which does not allow any ATS to be started. After the first restart, any subsequent restarts set the ATP to “Idle” which does allow ATSs to be started. I also found that issuing a StopATS command sets the ATP to “Idle”.
GSFCCFS-1657	Type mismatch of AtsCmdIndexBuffer	The AtsCmdIndexBuffer is defined as an array of int32 in sc_app.h. However, the table associated with this (CMD Status) is defined as an array of uint8. This causes the status of command #1 to be placed in the table at #4, #2 is at #8, etc.. The AtsCmdIndexBuffer is defined in the AppData_t structure. However, in the sc_app.c, this is allocated as an array of uint8 (Line 413)

GSFCCFS-1663	SC app sources do not include sc_verify.h	The sc_verify.h header file is not being included in sc_app.c or any other source files. Consequently it is not being compiled and no platform configuration values are being verified. When sc_verify.h is added to sc_app.c, several values defined in sc_platform_cfg.h cause verification errors. The erroneous values and the corresponding verification macros need to be checked and updated appropriately.
GSFCCFS-1673	SC memcpy sizing issue	JSC is working on getting SC integrated into their setup. They are running into an issue where the first command from an RTS is sent correctly, but no subsequent commands get sent. The RTS validates correctly and claims that it finishes processing. There appears to be a difference in the way that the validate function parses the table vs the way that the actual processing function does. In the validate function: https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_sc/-/blob/GSFCCFS-1657/fsw/src/sc_loads.c#L435 In the sender: https://aetd-git.gsfc.nasa.gov/gsfc-cfs/cfs_sc/-/blob/develop/fsw/src/sc_state.c#L188
GSFCCFS-1692	SC: investigate RTS start behavior with invalid table size	The RTS buffer size, configured via sc_platform_cfg.h SC_RTS_BUF_SIZE, must accommodate the RTS table size in order to pass table validation. A table that exceeded the RTS buffer size was still allowed to be started via the RTS start command. An investigation of why the RTS was allowed to start is required in order to determine if there is an error handling defect, operator error, or if this is expected behavior.
GSFCCFS-1693	Add utility functions for index to ATS/RTS ID	Throughout SC there's frequent use of +1/-1 adjustments to convert between array index (zero-based) and ATS/RTS ID (one-based). While the current behavior is correct, utility functions could be used for this translation to improve code clarity/readability.
GSFCCFS-1712	SC should not pend forever on the software bus	
GSFCCFS-1716	SC should validate ATS index and ID before conversion and array use	There are several instances in SC where the ATS index and ID are not being validated before use. This validation should occur before being converted using the macros or before being used as an index into an array. Not every use of ATS index and ID requires this because the validation can occur before the ID value is stored in the ATS control block. Many functions use the ATS ID after retrieving it from the ATS control block. Whenever the ATS ID is taken directly from a command buffer, it must be validated.
GSFCCFS-1722	Default RTS tables fail validation	
GSFCCFS-1723	Default ATS tables fail validation	The RTS enable and start commands defined in the default ATS table need 2 byte padding.

GSFCCFS-1725	SC ATS table format is not robust	The use of a uint32 time tag in the ATS requires changes to tables on big-endian vs little-endian platforms above and beyond the use of the MAKE_BIG macro.

1.3 MISSING PLANNED FEATURES AND KNOWN PROBLEMS

Table 1.3-1 identifies currently open DCRs that are not addressed in this build. Any workarounds that may apply are identified.

Information on currently open DCRs is available at:

https://etdjira.gsfc.nasa.gov/projects/GSFCCFS/issues

Note that this is a restricted website that requires a server account. Additional DCRs may have been submitted after preparation of this VDD. A cFS SC DCR report containing a listing of open DCRs is available upon request for customers who do not have access to the restricted server. Please contact the cFS Program Team, cfs-program@nasa.onmicrosoft.com.

Table 1.3-1 – Currently open DCRs

Key	Summary	Description
GSFCCFS-1445	Consolidate common patterns in SC	Could reduce lines of code by using a loop for several common patterns in SC initialization. Specifically the sequence of CFE_TBL_Register calls in SC_RegisterAllTables(), the sequence of CFE_TBL_GetAddress calls in SC_GetDumpTablePointers(), and the sequence of CFE_TBL_NotifyByMessage calls in SC_RegisterManageCmds().
GSFCCFS-1447	Use OS_stat to verify file existence	In SC_LoadDefaultTables, the OS_OpenCreate function is used to verify that a file can be opened before any attempt is made to load a table from it. Since the code as-is appears to only be checking for file existence (and not performing any validation on the files), code could be streamlined by using OS_stat.
GSFCCFS-1453	Clarify code by making array index 0 reserved or unused	The command interface to SC identifies RTSs and ATSs starting with 1, but the code identifies them starting with 0. Thus in each command, there is code to adjust the Table Number to the Table Index. Code could be simplified by making array index 0 reserved or unused so that the conversion could be avoided.
GSFCCFS-1458	Consolidate similar functions	SC_ManageAtsTable, SC_ManageRtsTable, and SC_ManageAppendTable are similar

		enough that some consolidation may be possible.
GSFCCFS-1460	Reorder conditions to optimize code	In SC_ProcessAtpCmd, the following condition could be reordered: if ((!SC_CompareAbsTime (SC_AppData.NextCmdTime[SC_ATP], SC_AppData.CurrentTime)) && (SC_AppData.NextProcNumber == SC_ATP) && (SC_OperData.AtsCtrlBlckAddr -> AtpState == SC_EXECUTING)) Since SC_CompareAbsTime() is more complex than the next two conditions, put it as the last one to save CPU cycles should any of the simple comparisons fail Another small optimization: the "StillProcessing = false" condition could be moved out of the if and else blocks starting at line 201 in the SC_LoadAts function
GSFCCFS-1461	SC_ProcessAtpCmd is very long - could be refactored	
GSFCCFS-1466	Combine loops in SC_BuildTimeIndexTable	There are two loops in SC_BuildTimeIndexTable that appear to be easily combinable.
GSFCCFS-1467	Add a ContinueRtsOnFailureFlag to SC	There is currently a "ContinueAtsOnFailureFlag" that allows an operator to choose whether the processing of an ATS should continue after an error has been encountered. There is currently no corresponding option for RTS processing.
GSFCCFS-1468	Allow RtsGrp commands to accept non-contiguous ranges	Currently the RtsGrp commands accept a range of RTSs from start_index to end_index. Instead, the command could accept a set of bytes where each bit corresponds to the RTS to start. This would allow the group commands to work with a range of RTSs that is non-sequential
GSFCCFS-1471	Streamline SC_GetNextRtsCommand and SC_ParseRts	Both functions use temporary buffers and potentially unnecessary copies.
GSFCCFS-1645	SC Code Simplification	SC could use significant code refactoring and simplification. It would make code more readable, but could also reduce the number of copies required to read and send the software bus messages.
GSFCCFS-1717	SC untestable branch condition	SC has an untestable branch condition in sc_state.c SC_UpdateNextTime. The function checks the RtsNumber using <= SC_NUMBER_OF_RTS. However, the sub function called before the check, SC_GetNextRtsTime, guarantees that the RtsNumber cannot ever exceed SC_NUMBER_OF_RTS.

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	3.1.0	https://github.com/nasa/SC
Doxygen Documentation	Yes	N/A	https://github.com/nasa/SC
Unit Test Data	Yes	3.1.0	https://github.com/nasa/SC
FSW Make Files	Yes	3.1.0	https://github.com/nasa/SC

3.0 INSTALLATION PROCEDURES

In order to build and install the SC 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 sc)
SET(TGT1_FILELIST cfe_es_startup.scr)
```

After SC 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/master/cmake/README.md

4.0 CONFIGURATION SUMMARY AND VERSION IDENTIFICATION

This software can be found in the SC GitHub repository (https://github.com/nasa/SC) under the tag “3.1.0”.

Verification of the version can be done by sending an SC 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
COTS ..... Commercial Off-The-Shelf
CPU ..... Central Processing Unit
DCR ..... Discrepancy/Change Request
ETU ..... Engineering Test Unit
FSB ..... Flight Software Branch
FSW ..... Flight Software
GSFC ..... Goddard Space Flight Center
I&T ..... Integration & Test
JSC ..... Johnson Space Center
POSIX ..... Portable Operating System Interface
RTOS ..... Real-Time Operating System
SC ..... Stored Commands
SMP ..... Symmetric Multiprocessing
T&C ..... Telemetry and Command
TBD ..... To Be Determined
URL ..... Universal Resource Locator
VDD ..... Version Description Document