Code 582
Flight Software Branch

CORE FLIGHT SYSTEM
Data Storage
BUILD 2.5.1.0

FLIGHT SOFTWARE BUILD VERIFICATON TEST REPORT

Flight Software Branch – Code 582

Version 1.0

SIGNATURES

Submitted by:

X WALTER MOLESKI

Digitally signed by WALTER MOLESKI
DN: c=US, o=U.S. Government, ou=NASA, ou=People, cn=WALTER MOLESKI, 0.9.2342.19200300.100.1.1=wmoleski
Date: 2017.02.14 07:27:31 -05'00'

Walt Moleski/582
cFS Flight Software Tester

Approved by:

X SUSANNE STREGE

Digitally signed by SUSANNE STREGE
DN: c=US, o=U.S. Government, ou=NASA, ou=People, cn=SUSANNE STREGE, 0.9.2342.19200300.100.1.1=sstrege
Date: 2017.02.14 07:16:55 -05'00'

Susanne Strege/582
cFS Flight Software Product Development Lead

PLAN UPDATE HISTORY

Version	Date	Description	Affected Pages
1.0	02/07/2017	Initial Release	All

TABLE OF CONTENTS

1 INTRODUCTION

1.1 DOCUMENT PURPOSE

This Test Report describes the test results from the core Flight System (cFS) Data Storage (DS) Flight Software (FSW) Test Team builds 2.5.0.0 and 2.5.1.0 verification testing. The DS 2.5.1 build contains the changes associated with builds 2.5.0.0 and 2.5.1.0. The 2.5.1.0 revision includes a bug fix found during DS build 2.5.0.0 Build Verification Testing (BVT). Build 2.5.1.0 is the final build of the cFS DS application following BVT of builds 2.5.0.0 and 2.5.1.0 of the cFS DS application.

BVT is used to verify that the DS FSW has been tested in a manner that validates that it satisfies the functional and performance requirements defined within the cFS DS Requirements Document. This Test Report summarizes the FSW test history, the build verification process, the build test configuration, and the test execution and results.

1.2 APPLICABLE DOCUMENTS

Unless otherwise stated, these documents refer to the latest version.

Parent Documents (Mission and FSW)

- 582-2008-008 cFS Data Storage Requirements Document, Version 1.3
- 582-2008-012 cFS Deployment Guide, Version 3.0

Reference Documents

All of the references below can be found on the Code 582 internal website at http://fsw.gsfc.nasa.gov/

- 582-2003-001 FSB FSW Test Plan Template
- 582-2004-001 FSB FSW Test Description Template
- 582-2004-002 FSB FSW Test Scenario Template
- 582-2004-003 FSB FSW Test Procedure Template
- 582-2004-004 FSB FSW Test Execution Summary Template
- 582-2004-005 FSB Test Product Peer Review Form
- 582-2000-002 FSB FSW Unit Test Standard

1.3 DOCUMENT ORGANIZATION

Section 1 of this document presents some introductory material.

Section 2 provides a flight software overview and context along with the test history and testing overview.

Section 3 describes the build verification process including procedure development and execution and test products produced.

Section 4 describes the build test configuration which includes an overview of the testbed and the requirements verification matrix.

Section 5 describes the test execution and results by subsystem.

Appendix A - provides the Requirements Traceability Matrix

Appendix B - provides the Command, Telemetry, and Events Verification Matrix

1.4 DEFINITIONS

There were 3 verification methods used during build verification testing. They were:

- Demonstration: Show compliance with system requirement by exhibiting the required capability (e.g. by demonstrating interactive capability, display capability, print capability, etc.
- Inspection: Show compliance with a system requirement by visual verification of the software (e.g. verifying preparation for delivery, proper interfacing)
- Analysis: Perform detailed analysis of code, generated data (both intermediate data and final output data), etc., to determine compliance with system requirements.

The fields in the Requirements Verification Matrix in Section 4.3 are defined as follows:

- Requirements Tested Passed: Requirement was fully tested in a build test procedure and passed all tests.
- Requirements Tested Failed: Requirement was fully tested in a build test procedure and failed one or more aspect of the testing.
- Requirements Tested Partially: Requirement was tested partially in a build test procedure. To be fully tested, the partially tested requirement is either tested additionally in one or more other test procedures within the same build **and/or** other aspects of the requirement must be tested in a later build, due to capabilities not present in the current build
- Total Tested: Total number of requirements fully tested in a build test procedure. Includes total passed and total failed, but does **not** include requirements tested partially, **unless** (included as a separate entry) testing in multiple procedures within the same build constitutes total testing of a particular requirement. Total Requirements Tested is computed this way in order to avoid multiple counting of individual requirements that are tested partially in more than one procedure.
- Deferred: Number of requirements that were planned to be tested in current build, but were not tested due to some FSW capability or necessary system component not being present.
- Total: Total Requirements Tested + Number of Requirements Deferred

In each software test section in Section 5 there is a table of DCR's. The state definitions are as follows:

- Opened: The DCR is currently being addressed
- Assigned: The DCR was accepted and the modification is being addressed
- InTest: The DCR was corrected and is currently in test
- Validated: The DCR was corrected and tested and has been validated, needs to have a CCB to close the DCR
- Closed: The DCR is closed and have been resolved and tested to satisfaction
- Closed with Defect: The DCR is closed and the defect is most likely assigned a differed DCR number associated with another subsystem.

2 OVERVIEW

2.1 FLIGHT DATA SYSTEM CONTEXT

Figure 2-1 illustrates the cFS system context. The core Flight Executive (cFE) interfaces to five external systems: an Operating System (OS), a Hardware Platform (HP), an Operational Interface (OI), Applications (APP), and other cFE-based systems.

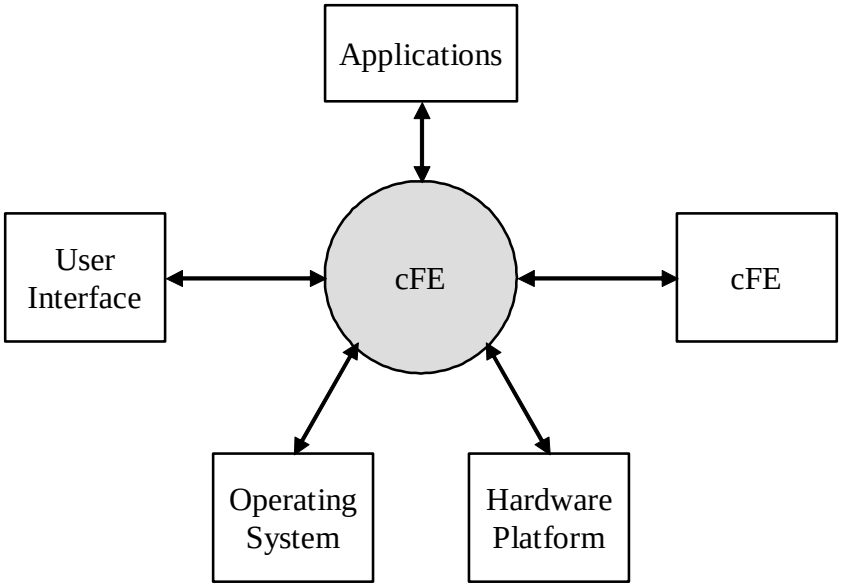

Figure 2-1 cFS System Context

Figure 2-2 below shows context diagram for the cFS Data Storage (DS) Application. During initialization, DS subscribes to messages from other applications as defined in the Data Storage Filter Table. The Scheduler Application (SCH) sends periodic commands to DS as defined in the SCH Schedule Table which requests housekeeping from DS (note that a mission must define this request in the SCH table). Ground commands come from the Command Ingest task (CI). Messages are routed to DS by the cFE Software Bus (SB) Application. DS learns of ground updates to the DS tables through the cFE Table Services application.

DS defines 2 tables:

1. Filter Table – for each message ID, specifies filter scheme, filter parameters and which file or files to write the message to. Note that each message can be written to a <PLATFORM_DEFINED> number of destination files.

2. File Table – specifies information about each destination file.

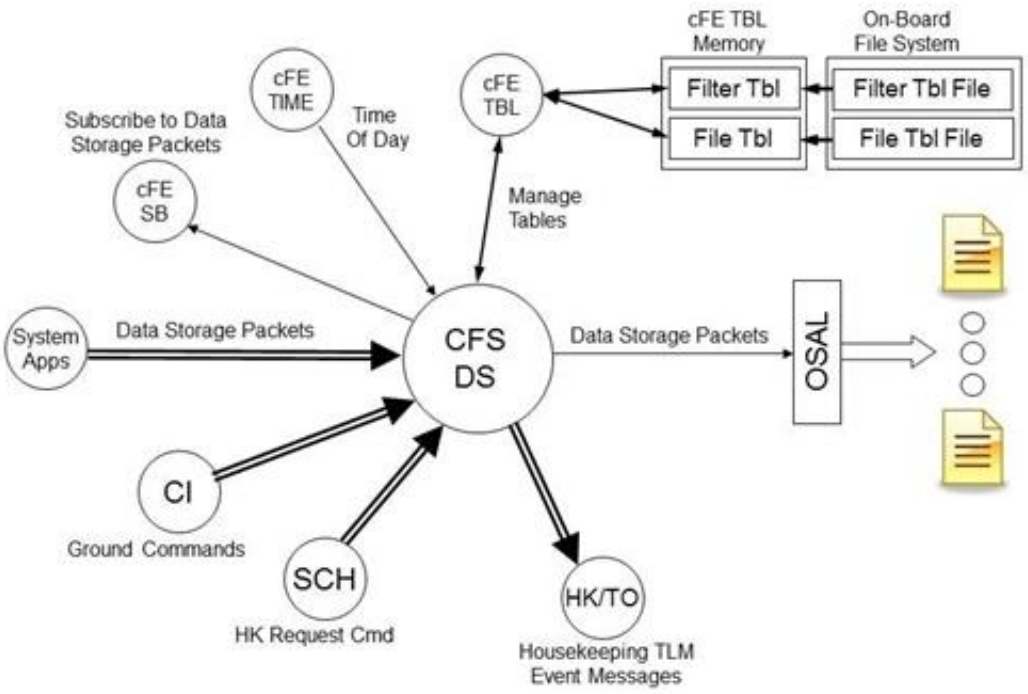

Figure 2-2 cFS DS Context

2.2 TEST HISTORY

DS 2.0.0.0 – Build Verification Testing completed 12/2/2009 by Walt Moleski
DS 2.1.1.0 – Build Verification Testing completed 1/5/2011 by Walt Moleski
DS 2.2.0.0 – Build Verification Testing completed 9/19/2011 by Walt Moleski
DS 2.3.0.0 – Build Verification Testing completed 7/31/2012 by Walt Moleski
DS 2.4.0.0 – Build Verification Testing completed 1/23/2015 by Walt Moleski
DS 2.4.1.0 – Build Verification Testing completed 7/30/2015 by Walt Moleski
DS 2.5.0.0 – Build Verification Testing completed 2/3/2017 by Walt Moleski
DS 2.5.1.0 – Build Verification Testing completed 2/7/2017 by Walt Moleski

2.3 TESTING OVERVIEW

The cFS Test procedures assume that the cFS application and its corresponding test application are not executing before the start of the test. If this is the case, the test procedures will need to be modified to handle this situation.

The DS application was tested during Build Verification testing using the following:

- 1 test application: tst_ds
- 7 main test procedures: ds_filewrite.prc, ds_filter.prc, ds_gencmds.prc, ds_movefile.prc, ds_resetcds.prc, ds_resetnocds.prc; ds_enablestate.prc
- 7 procedures that setup the Packet Filter and Destination File Tables: ds_tbl1.prc, ds_tbl2.prc, ds_tbl3.prc, ds_tbl4.prc, ds_tbl5.prc, ds_badfiletbls.prc, and ds_badfiltertbls.prc
- 1 procedure called by the main procedures to startup the DS and TST_DS applications: ds_start_apps.prc
- All tests require the ASIST Ground Station

The TST_DS test application is used to send schedule requests for the output of DS's housekeeping data to the DS application. This was useful when performing build verification testing since it provided great control over the sequence of steps. In addition, having the test application eliminated the need to modify the SCH_LAB application and rebuild. When deployed for a mission, the Scheduler Application would provide this request. In addition, the test application has 5 ground commands defined to help with the DS testing. These commands are described below:

- TST_DS_NOOP
  - This command that issues an event and increments the command processed counter.
- TST_DS_ResetCtrs
  - This command resets the command processed and command error counters to zero (0).
- TST_DS_SetCounters
  - This command sets several Data Storage (DS) counters so that the DS_ResetCtrs command can be tested and verified.
- TST_DS_SendMessage
  - This command constructs a cFE Software Bus (SB) message using the supplied arguments. The arguments are the Message ID to send, the Message Type which determines the amount of data contained in the message and the data pattern for the data.
- TST_DS_SetSequenceCnt
  - This command sets the next sequence count in the SB message to the supplied count for the specified message type.
- TST_DS_SendTimeMessage
  - This command constructs a SB message using the supplied arguments as the message timestamp. The arguments are the Message ID to send which must be a TLM message since the timestamp is in the secondary header, the Message Type which determines the amount of data contained in the message, the data pattern for the data and the seconds and subseconds to use as the message timestamp.

The main DS test procedures do the following:

Procedure	**Description**
ds_enablestate	This test verifies that the cFS Data Storage (DS) application starts up in the proper state based upon the DS_DEF_ENABLE_STATE and DS_CDS_ENABLE_STATE configuration parameter settings. This test procedure was created for DS 2.5.1.0.
ds_filewrite	This test verifies that the cFS Data Storage (DS) application writes messages to files according to the requirements. Also, this test verifies the DS application commands and that DS handles anomalies appropriately.
ds_filter	This test verifies that the cFS Data Storage (DS) application filters messages according to the requirements. This test also verifies that the DS application handles anomalies appropriately.
ds_gencmds	This test verifies that the cFS Data Storage (DS) general commands function properly. The NOOP and Reset Counters commands will be tested. Invalid versions of these commands will also be tested to ensure that the DS application handled these properly.
ds_movefile	This test verifies that the cFS Data Storage (DS) application writes messages to files according to the requirements and moves these files to the table-specified directory after they are closed. **NOTE:** This test SHOULD NOT be executed if the Move Files configuration parameter is set to FALSE by the Mission.

Procedure	Description
ds_resetcds	This test verifies that the cFS Data Storage (DS) application initializes the appropriate data items based upon the type of initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This test also verifies that the proper notifications occur if any anomalies exist with the data items stated in the requirements. **NOTE:** This test SHOULD NOT be executed if the Make Tables Critical configuration parameter is set to 0 by the Mission.
ds_resetnocds	This test verifies that the cFS Data Storage (DS) application initializes the appropriate data items based upon the type of initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This test also verifies that the proper notifications occur if any anomalies exist with the data items stated in the requirements. **NOTE:** This test SHOULD NOT be executed if the Make Tables Critical configuration parameter is set to 1 by the Mission.

The test procedures described in the table below are called by at least one of the test procedures above.

Procedure	Description
ds_badfiletbls	This procedure creates seven Destination File Table load image files that each contain one of seven different errors that are checked by the table validation function. These files are used in the ds_resetnocds test.
ds_badfiltertbls	This procedure creates five Packet Filter Table load image files that each contain one of five different errors that are checked by the table validation function. These files are used in the ds_resetnocds test.
ds_start_apps	This procedure performs all the necessary actions to start the DS and TST_DS applications if they are not currently executing.
ds_tbl1	This procedure creates the initial Destination File and Packet Filter table load images for the ds_gencmds test.
ds_tbl2	This procedure creates the initial Destination File and Packet Filter table load images for the ds_filewrite test.
ds_tbl3	This procedure creates the initial Destination File and Packet Filter table load images for the ds_filter, ds_resetcds, and ds_resetnocds tests.
ds_tbl4	This procedure creates the initial Destination File and Packet Filter table load images for the ds_movefile test.
ds_tbl5	This creates a full cFS Data Storage (DS) Packet Filter Table load image file.

The cFS Deployment Guide contains the instruction for how to set up both the cFS Flight and Ground test environment. The testers use a cFS Test Account for each build test. This account runs the Advanced Spacecraft Integration and System Test Software (ASIST) and is setup to contain all the files needed to test the application. These files are extracted from MKS, the source repository tool. Included in these files are test utilities. These utilities can be located in 2 places depending upon whether they are “local” or “global” utilities. The local utilities are extracted into the working prc directory ($WORK/prc). The global utilities are pointed to by ASIST in the global area defined on the test system. Additional tools utilized by the test procedures are located in the $TOOLS directory. It is assumed that test procedures and the ASIST telemetry database used for testing is built using procedure and database templates.

The following utilities were used during testing:

Name	Description
CFE_startup	Directive combines the "start_data_center", "open_tlm", and "open cmd <cpu>" ASIST startup commands.
close_data_center	Directive that closes the command and telemetry connection to the CPU being used.
create_tbl_file_from_cvt	Procedure that creates a load file from the specified arguments and cvt

Name	Description
load_start_app	Procedure to load and start a user application from the /s/opr/accounts/cfstest/apps/cpux directory.
load_table	Procedure that takes the specified file and transfers the file to the specified processor and then issues a TBL_LOAD command using the file.
tst_ds (version 2.5.1.0)	Test application required to test the DS application.
ut_pfindicate	Directive to print the pass fail status of a particular requirement number.
ut_runproc	Directive to formally run the procedure and capture the log file.
ut_sendcmd	Directive to send EVS commands. Verifies command processed and command error counters.
ut_sendrawcmd	Send raw commands to the spacecraft. Verifies command processed and command error counters.
ut_setrequirements	A directive to set the status of the cFE requirements array.
ut_setupevents	Directive to look for multiple events and increment a value for each event to indicate receipt.
ut_tlmwait	Directive that waits for the specified telemetry condition to be met
ds_file_util	This is the executable of the ds_file_util.c program that takes a binary data file as its input and generates two output files. The only pertinent output file is the xxx_raw.txt file where xxx is the original binary filename.

2.4 VERSION INFORMATION

Item	Version
DS Requirements	1.3
DS Application	2.5.1.0
TST_DS Application	2.5.1.0
CFE	6.5.0.0
OSAL	4.2.0.0
ASIST	20.2
VxWorks	6.9

3 BUILD VERIFICATION TEST PREPARATION

3.1 SCENERIO DEVELOPMENT

There were no new scenarios developed for DS 2.5.0.0 or 2.5.1.0 Build Verification Test. All scenarios are stored on the MKS server, in cFS-Repository DS test-and-ground directory within the Scenarios subdirectory. It should be noted that as DS requirements evolve these scenarios are not updated to reflect any changes made.

3.2 PROCEDURE DEVELOPMENT AND EXECUTION

This build test was completed by running 7 test procedures. The ds_enablestate procedure was developed for DS 2.5.1.0 in order to test the DS_DEF_ENABLE_STATE and DS_CDS_ENABLE_STATE configuration parameters more thoroughly. This new procedure was created since these parameters were originally tested in the ds_resetcds and ds_resetnocds in DS 2.4.1.0 and do not have any relationship with the DS_MAKE_TABLES_CRITICAL configuration parameter that is tested by these procedures. All test procedures were written using the STOL scripting language. The naming convention for files created by the test procedures was: scx_cpu<#>_<procedure name>_GMT.<ext>.

3.3 TEST PRODUCTS

Five log files were generated for every procedure that was run. They are defined as follows:

- Logs with the .loge extension list all events sent by the flight software
- Logs with the .logr extension list all requirements that passed validation by demonstration
- Logs with the .logp extension lists all prints that are generated by the test procedure
- Logs with the .logf extension lists everything from the other logs along with the steps in the test procedure
- Logs with the .logs extension lists the SFDU information (if applicable) contained in the full log.

A test summary report is developed in MKS for each procedure by the tester after build testing is completed. All test products are maintained on MKS in the cFS-Repository DS test-and-ground directory.

4 BUILD VERIFICATION TEST EXECUTION

4.1 TESTBED OVERVIEW

DS FSW testing took place in the cFS FSW Development and Test Facility. A high level view of the cFS FSW Test Bed is shown in Figure 4-1. This facility is located in GSFC Building 23, Room N410. This facility consists of two ASIST workstations running ASIST version 20.2 and three MPC750 CPU boards running VxWorks 6.4. CPU1 is primarily used for development testing while CPU2 and CPU3 are used for build verification testing.

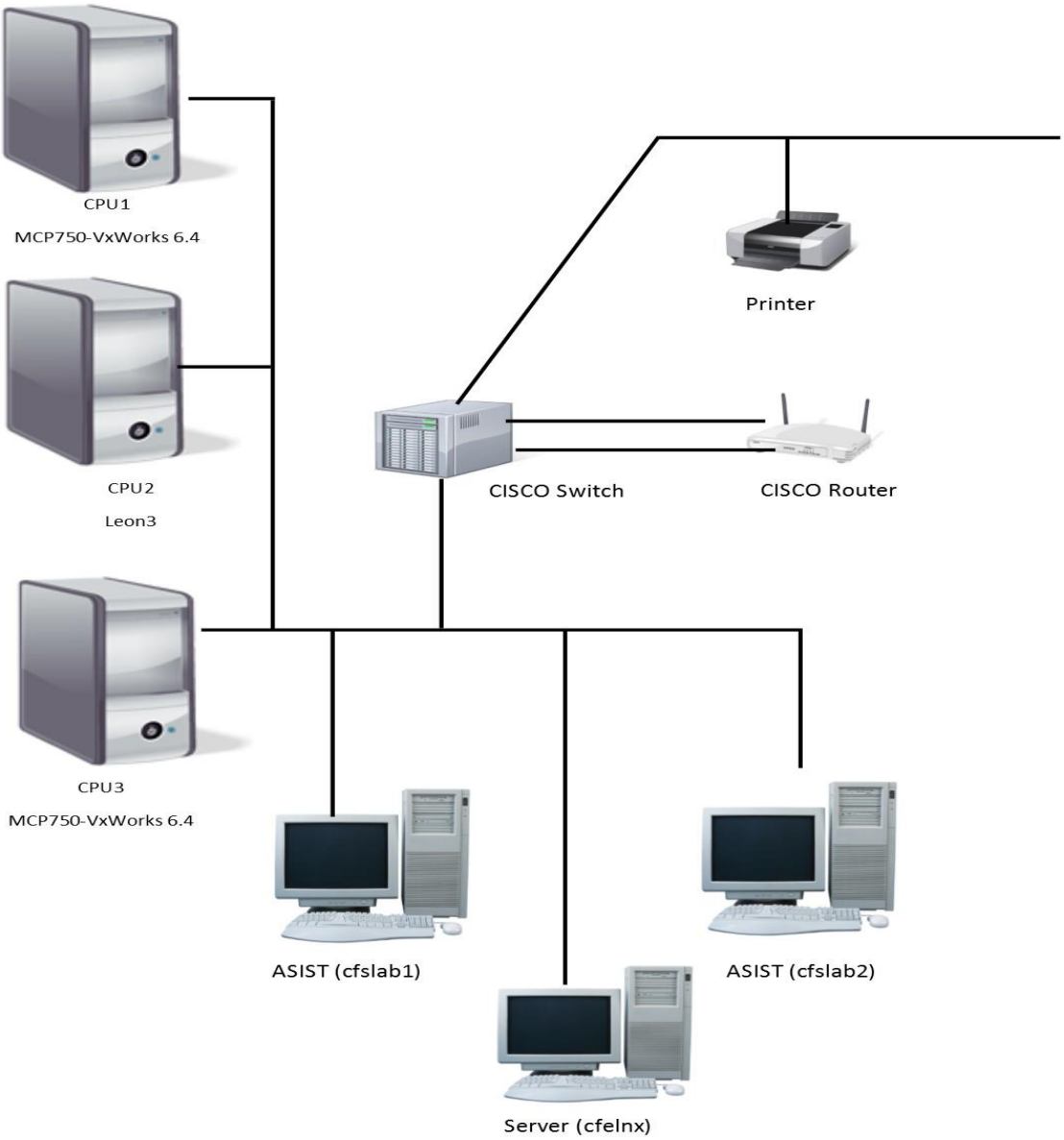

Figure 4-1 cFS FSW Development and Testing Facility

4.2 REQUIREMENTS VERIFICATION MATRIX

	Data Storage (DS)
Requirements Tested Passed	58
Requirements Tested Failed	0
Requirements Tested Partially	0
Total Tested	58
Deferred	0
Untestable	0
Total	58

4.3 REQUIREMENTS PARTIALLY TESTED

There were no requirements that were partially tested.

4.4 REQUIREMENTS/FUNCTIONALITY DEFERRED

No requirements were deferred to later build testing.

4.5 REQUIREMENTS/FUNCTIONALITY DEFERRED FOR MISSION TESTING

No requirements/functionality was deferred to mission testing:

5 BUILD VERIFICATION TEST RESULTS

5.1 OVERALL ASSESSMENT

During this build test of the DS Application, the software behaved as expected.

Below is a summary of the results:

- 47 requirements passed demonstration
- 11 requirements passed via analysis
- 8 DCRs were verified

5.2 PROCEDURE DESCRIPTION

Procedure	Description	Requirements tested
ds_enablestate	This test verifies that the cFS Data Storage (DS) application starts up in the proper state based upon the DS_DEF_ENABLE_STATE and DS_CDS_ENABLE_STATE configuration parameter settings. This test procedure was created for DS 2.5.1.0.	DS8000; DS9000; DS9003; DS9004; DS9008
ds_filewrite	This test verifies that the cFS Data Storage (DS) application writes messages to files according to the requirements. Also, this test verifies the DS application commands and that DS handles anomalies appropriately.	DS1002; DS1004; DS1005; DS3000; DS3000.1; DS3000.1.1; DS3000.2; DS3000.2.1; DS3001; DS3001.1; DS3001.2; DS3002; DS3002.1; DS3003; DS3004; DS3005; DS5000; DS5001; DS5002; DS5003; DS5004; DS5005; DS5006; DS5011; DS5012; DS5013; DS5014; DS5015; DS8000; DS9000
ds_filter	This test verifies that the cFS Data Storage (DS) application filters messages according to the requirements. This test also verifies that the DS application handles anomalies appropriately.	DS1002; DS1004; DS1005; DS2000; DS2000.1; DS2000.2; DS2001; DS2001.1; DS2002; DS2002.1; DS2003; DS2003.1; DS2003.2; DS3000; DS3000.1; DS3000.1.1; DS3000.2; DS3000.2.1; DS3003; DS3004; DS5002; DS5008; DS5009; DS5010; DS5016; DS5016.1; DS5016.2; DS5016.3; DS5016.4; DS8000; DS9000
ds_gencmds	This test verifies that the cFS Data Storage (DS) general commands function properly. The NOOP and Reset Counters commands will be tested. Invalid versions of these commands will also be tested to ensure that the DS application handled these properly.	DS1000; DS1001; DS1002; DS1004; DS1005; DS8000; DS9000

Procedure	Description	Requirements tested
ds_movefile	This test verifies that the cFS Data Storage (DS) application writes messages to files according to the requirements and moves these files to the table-specified directory after they are closed. **NOTE:** This test SHOULD NOT be executed if the Move Files configuration parameter is set to FALSE by the Mission.	DS1004; DS3000; DS3000.1; DS3000.1.1; DS3001; DS3001.1; DS3001.2; DS3002; DS3002.1; DS5002; DS8000; DS9000
ds_resetcds	This test verifies that the cFS Data Storage (DS) application initializes the appropriate data items based upon the type of initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This test also verifies that the proper notifications occur if any anomalies exist with the data items stated in the requirements. **NOTE:** This test SHOULD NOT be executed if the Make Tables Critical configuration parameter is set to 0 by the Mission.	DS8000; DS9000; DS9001; DS9002; DS9003; DS9005; DS9006; DS9007; DS9008
ds_resetnocds	This test verifies that the cFS Data Storage (DS) application initializes the appropriate data items based upon the type of initialization that occurs (Application Reset, Processor Reset, or Power-On Reset). This test also verifies that the proper notifications occur if any anomalies exist with the data items stated in the requirements. **NOTE:** This test SHOULD NOT be executed if the Make Tables Critical configuration parameter is set to 1 by the Mission.	DS8000; DS9000; DS9001; DS9002; DS9004; DS9005; DS9006; DS9007; DS9008

5.3 ANALYSIS REQUIREMENTS VERIFICATION

There were 11 requirements verified using analysis and are described below:

Requirement	Requirement Text	Analysis
DS2001	DS shall store "N of X" messages starting at offset O.	Three tests were performed for this requirement in the Filter test procedure. The tests were N=1, X=3 and O=1; N=2, X=3 and O=1; and N=4, X=6 and O=2. The correct numbers of packets were stored in each test. However, the third test required some additional analysis to make sure the correct packets were stored. The CCSDS sequence counts for the 6 packets sent were 4 - 9. The mod 6 values for these packets were 4, 5, 0, 1, 2, 3 respectively. So, starting at offset 2, the next 4 packets will be stored. Thus, packets 4, 5, 2 and 3 should have been stored which they were.
DS2003	The input argument to the filter algorithm shall be determined by the Filter Type indicator.	This requirement was passed in the Filter test procedure by sending a number of messages with Sequence based and Time based filtering.
DS2003.1	If Filter Type indicates Sequence based filtering then the algorithm input argument is a value equal to the packet sequence count.	This requirement passed in the Filter test procedure with Step 3.13. As documented above for DS2001, the correct packets were received based upon the filtering parameters and the sequence count of the packet.
DS2003.2	If Filter Type indicates Time based filtering then the algorithm input argument is a value created from the lower 11bits of packet timestamp seconds plus the high 4 bits of timestamp subseconds.	This requirement passed in the Filter test procedure with Step 3.16. As documented above for DS2001, the correct packets were received based upon the filtering parameters and the timestamp of the packet.
DS3000.1	If the Filename Type indicates naming based on Sequence, the constructed filename will include a character representation of the Sequence Count value from the Destination File Table.	This requirement passed in the FileWrite and Filter test procedures. The output captured in the logp files verify that the character representation of the sequence count is contained in the filename. The FileWrite steps are 2.1.3, 2.2, 2.3.2, 2.6, 2.7, 3.1.1, 3.6, 3.9 and 3.14. The Filter steps are 2.1, 2.4.2, 2.8, 2.17, 3.3, 3.8 and 3.13.

Requirement	Requirement Text	Analysis
DS3000.1.1	If the Filename Type indicates naming based on Sequence, the value of the Sequence Count shall be incremented each time a file is created.	This requirement passed in the FileWrite and Filter test procedures. The output contained in the logp files verify that the sequence count incremented when each file was created. The FileWrite steps are 2.2, 2.3.2, 2.6, 2.7, 3.6 and 3.9. The Filter steps are 2.4.2, 2.8, 2.17, 3.3, 3.8 and 3.13.
DS3000.2	If the Filename Type indicates naming based on Time, the constructed filename will include a character representation of the packet timestamp.	This requirement passed in the FileWrite and Filter test procedures. The output captured in the logp files verify that the packet timestamp was represented and formatted properly. The FileWrite step is 2.4.2 and the Filter steps are 2.2 and 2.14.
DS3000.2.1	If the Filename Type indicates naming based on Time, the file date and time shall be represented in the following format:"YYYDDDHHMMSS".	This requirement passed in the FileWrite and Filter test procedures. The output captured in the logp files verify that the packet timestamp was represented and formatted properly. The FileWrite step is 2.4.2 and the Filter steps are 2.2 and 2.14.
DS3001.2	A minimum of one packet shall be written to a file.	This requirement passed in the FileWrite test procedure. Step 2.3.2 sends a message with 1500 data bytes to a destination whose maximum size is set to 1024 bytes. The file was created and closed when the next message for this ID is received in Step 2.6. The file was analyzed and found to contain only the packet sent in Step 2.3.2
DS3003	Each DS destination file shall contain a primary cFE file header with the following information: a) Content Type b) SubType c) Primary header length d) Spacecraft ID e) Processor ID f) Application ID g) File creation time (seconds) h) File creation time (sub-seconds)	This requirement passed in the FileWrite and Filter test procedures. The contents of each file were verified and contained the items specified. There was an additional 32 -byte description in the Primary header.

Requirement	Requirement Text	Analysis
DS3004	Each DS destination file shall contain a secondary file header with the following information: a) File close time (seconds) b) File close time (sub-seconds) c) File table index d) Qualified filename	This requirement passed in the FileWrite and Filter test procedures. The contents of each file were verified and contained the items specified. There was an additional 2-byte file table type in the Secondary header.

5.4 DCRS

One new DCR was generated during DS 2.5.0.0 testing: DCR 146041

No DCRs were generated during DS 2.5.1.0 testing.

5.4.1 DCRs Verified

The following DCRs were verified during testing:

DCR	Description	Test Method	Test Approach
4126	DS – Sequence Counter Will Always Roll Over to Zero	Inspection	The code change for this DCR entailed setting the sequence counter to its initial starting value upon rollover rather than zero.
141398	Remove Unnecessary Lines From ds_cmds.c	Inspection	The stated lines were removed from the code.
141537	Move function prototype from .c file to .h file	Inspection	The stated prototype was moved.
141542	DS – Need to add to ds_app.h: #include “ds_appdefs.h”	Inspection	The stated line was added to the ds_app.h file.
145916	DS – Hash Table Bug	Test Procedure	The AddMID command is tested in the ds_filter test procedure. This command passed as well as commands to Set the various filter items (Type, FileIndex,Parameters) on the added MID. The Set commands were performed from the STOL command line and verified by dumping the Filter Table an checking the contents of the new MID.
145920	Integrate and Implement Babelfish Ticket Fixes	Test Procedure	Three fixes were identified in this DCR. Two of these were implemented and passed.
145921	DS – CFE_EVS_SendEvent Format Warnings	Test Procedure	No compiler warnings were generated when DS 2.5.1.0 was built.
146041	DS: Several commands that use the FileIndex argument are no longer 32-bit aligned	Test Procedure	The DS Build Verification Tests send all the command and verify they are sent properly. All commands successfully passed both nominal and error case tests.

5.4.2 Outstanding DCRs/Trac Tickets

Trac ticket references are proceeded with a '#' character.

DCR or Trac #	Description	State
#73	DS file header values should be big-endian. As with CCSDS, which is standardizing on big-endian for message headers, the fields in the DS file header (close time, FileTableIndex, FileNameType) should be stored in big-endian order.	Work Completed
#63	DS file header should include additional metadata. DS currently stores a number of fields in the DS file header (DS_FileHeader_t), namely the time the file was closed, the file name, the file table index, and the file name type. When reading DS-created files on other platforms with other configurations, it is possible to tease apart platform/mission-specific information but it would be easier to store the configuration in the header for easier analysis of DS files.	Work Completed
#71	DS should (optionally) add a timestamp for each packet stored. CCSDS telemetry packets include a timestamp in the CCSDS headers. Command packets, on the other hand, do not. Also, if CCSDS timestamps are generated by something other than the local CPU, the timestamp may reflect when the packet was generated but not when the packet was received/stored by DS. Thirdly, if the CCSDS timestamp is generated using a different clock that is not in sync, the timestamps may not coincide. This is particularly important in multi-CPU environments, such as when cFS busses are connected via SBN. This will particularly help with replay using the ds_replay application as the timestamps will accurately reflect when DS received the packets and will be in the correct order.	Work Completed
4113	DS - Add Trick Simulation Support (JSC Request)	On Hold
4123	Unsubscribing to Filter Messages May Cause Flooding of SB No Subscribers Event. When a new filter table is loaded, DS unsubscribes to all packets in the old table and then subscribes to all packets in the new table. This is true even if the message IDs in question are present in both the old and new filter tables. If packets are being generated at the time DS is unsubscribing and DS is the only application that has subscribed to a packet, the system could be flooded with SB no subscriber events.	On Hold
4193	When using a DS configuration with Critical Tables, the Filter Table Filename is cleared when DS is restarted after a Processor and Application Reset AND the table is restored from the CDS. This was not expected. Dumping the Table Registry after such a restart confirmed that the Last Loaded file is cleared from the Table Registry which could be the cause of this problem.	Submitted
145950	Add Global Tables and Initialize Tables In Unit Test setup function.	In Work

5.5 NOTES

The ds_enablestate test procedure was executed 4 times for each unique setting of the DS_DEF_ENABLE_STATE and DS_CDE_ENABLE_STATE configuration parameters (1-1 - default; 1-0; 0-1; 1-1).
The ds_movefile test procedure was executed with a version of the DS software that had the DS_MOVE_FILES configuration parameter set to TRUE. The default delivered with DS is FALSE. Also, the underlying rdl needed to change to contain the Move Directory specification contained in the Destination File table.

It should be noted that integration testing is the ultimate verification of the DS application's performance in a system-like scenario.

APPENDIX A - RTTM

The DS Build 2.5.1.0 RTTM can be found on the MKS server, in cFS-Repository DS test-and-ground directory results folder.

APPENDIX B - COMMAND, TELEMETRY, AND EVENTS VERIFICATION MATRIX

Command	Test Procedure(s)	Notes/Comments
DS_NOOP	GenCmds	
DS_ResetCtrs	GenCmds	
DS_Enable	FileWrite	
DS_Disable	FileWrite	
DS_SetFilterFile	Filter	
DS_SetFilterType	Filter	
DS_SetFilterParams	Filter, ResetCDS	
DS_SetFileType	FileWrite	
DS_EnableFile	FileWrite	
DS_DisableFile	FileWrite	
DS_SetPath	FileWrite, ResetCDS	
DS_SetBasename	FileWrite, ResetCDS	
DS_SetExtension	FileWrite	
DS_SetMaxFileSize	FileWrite	
DS_SetMaxFileAge	FileWrite	
DS_SetFileSeqCtr	FileWrite	
DS_CloseFile	FileWrite, Filter, MoveFile	
DS_GetFileInfo	All	
DS_AddMID	Filter	
DS_CloseAllFiles	FileWrite	

Telemetry	Test Procedure(s)	Notes/Comments
DS_CMDPC	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_CMDEC	FileWrite, Filter, GenCmds, MoveFile,ResetCDS, ResetNoCDS	
DS_DisabledPktCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_IgnoredPktCnt	GenCmds, ResetNoCDS	
DS_FilteredPktCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_PassedPktCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_FileWriteCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_FileWriteErrCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_FileUpdCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_FileUpdErrCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_DestLoadCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_DestPtrErrCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_FilterLoadCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	

DS_FilterPtrErrCnt	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
DS_AppEnaState	FileWrite	
DS_FileState[].FileAge	FileWrite, MoveFile	
DS_FileState[].FileSize	FileWrite	
DS_FileState[].FileRate		
DS_FileState[].FileSeq		
DS_FileState[].EnableState	FileWrite	
DS_FileState[].OpenState	FileWrite, MoveFile	
DS_FileState[].FileName	FileWrite, Filter, MoveFile, ResetCDS, ResetNoCDS	
Table Telemetry		
DS_DF_TBL_Description	BadFileTbls, Tbl1, Tbl2, Tbl3, Tbl4	Table Load setup procedures
DS_DF_TBL[].MoveDirName	MoveFile, Tbl4	
DS_DF_TBL[].Pathname	FileWrite, MoveFile, ResetCDS	
DS_DF_TBL[].Basename	FileWrite, ResetCDS	
DS_DF_TBL[].Extension	FileWrite	
DS_DF_TBL[].FileNameType	FileWrite, Filter, MoveFile, ResetCDS, ResetNoCDS	
DS_DF_TBL[].FileState	FileWrite	
DS_DF_TBL[].FileSize	FileWrite	
DS_DF_TBL[].FileAge	FileWrite, MoveFile	
DS_DF_TBL[].SeqCnt	FileWrite	
DS_PF_TBL_Description	BadFilterTbls, Tbl1, Tbl2, Tbl3, Tbl4, Tbl5	Table Load setup procedures
DS_PF_TBL[].Index	FileWrite, Filter, MoveFile	
DS_PF_TBL[].FilterType	Filter	
DS_PF_TBL[].N_Value	Filter, ResetCDS	
DS_PF_TBL[].X_Value	Filter, ResetCDS	
DS_PF_TBL[].O_Value	Filter, ResetCDS	

Id	Event Message	Test Procedure(s)	Notes/Comments
1	DS_INIT_EID	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
2	DS_INIT_ERR_EID		
3	DS_EXIT_ERR_EID		
6	DS_INIT_CDS_ERR_EID		
7	DS_INIT_TBL_CDS_EID		
8	DS_INIT_TBL_ERR_EID	Filter, ResetCDS, ResetNoCDS	
10	DS_FIL_TBL_EID	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
11	DS_FIL_TBL_ERR_EID	ResetNoCDS	
12	DS_FLT_TBL_EID	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
13	DS_FLT_TBL_ERR_EID	ResetNoCDS	
14	DS_FILE_NAME_ERR_EID		
15	DS_CREATE_FILE_ERR_EID	FileWrite	
16	DS_WRITE_FILE_ERR_EID		
21	DS_CMD_CODE_ERR_EID	GenCmds	

22	DS_HK_REQUEST_ERR_EID	GenCmds	
31	DS_NOOP_CMD_EID	GenCmds	
32	DS_NOOP_CMD_ERR_EID	GenCmds	
33	DS_RESET_CMD_EID	GenCmds	
34	DS_RESET_CMD_ERR_EID	GenCmds	
35	DS_ENADIS_CMD_EID	FileWrite	
36	DS_ENADIS_CMD_ERR_EID	FileWrite	
37	DS_FILE_CMD_EID	Filter	
38	DS_FILE_CMD_ERR_EID	Filter	
39	DS_FTYPE_CMD_EID	Filter	
40	DS_FTYPE_CMD_ERR_EID	Filter	
41	DS_PARMS_CMD_EID	Filter, ResetCDS	
42	DS_PARMS_CMD_ERR_EID	Filter	
43	DS_NTYPE_CMD_EID	FileWrite	
44	DS_NTYPE_CMD_ERR_EID	FileWrite	
45	DS_STATE_CMD_EID	FileWrite	
46	DS_STATE_CMD_ERR_EID	FileWrite	
47	DS_PATH_CMD_EID	FileWrite, ResetCDS	
48	DS_PATH_CMD_ERR_EID	FileWrite	
49	DS_BASE_CMD_EID	FileWrite, ResetCDS	
50	DS_BASE_CMD_ERR_EID	FileWrite	
51	DS_EXT_CMD_EID	FileWrite	
52	DS_EXT_CMD_ERR_EID	FileWrite	
53	DS_SIZE_CMD_EID	FileWrite	
54	DS_SIZE_CMD_ERR_EID	FileWrite	
55	DS_AGE_CMD_EID	FileWrite	
56	DS_AGE_CMD_ERR_EID	FileWrite	
57	DS_SEQ_CMD_EID	FileWrite	
58	DS_SEQ_CMD_ERR_EID	FileWrite	
59	DS_CLOSE_CMD_EID	FileWrite, Filter, MoveFile	
60	DS_CLOSE_CMD_ERR_EID	FileWrite	
61	DS_MOVE_FILE_ERR_EID		
62	DS_GET_FILE_INFO_CMD_EID	FileWrite, Filter, GenCmds, MoveFile, ResetCDS, ResetNoCDS	
63	DS_GET_FILE_INFO_CMD_ERR_EID		
64	DS_ADD_MID_CMD_EID	Filter	
65	DS_ADD_MID_CMD_ERR_EID	Filter	
66	DS_CLOSE_ALL_CMD_EID	FileWrite	
67	DS_CLOSE_ALL_CMD_ERR_EID	FileWrite	