

12 - AxPAC cFS Developer's Guide

- [0. Prerequisites](#)
- [1. Setup Project Repository](#)
 - [1.1 Complete Setup](#)
 - [1.2 Verify_Setup_is_Correct](#)
 - [1.3 Execute_FSW_with_Ground_System_Tool](#)
 - [1.4 Create Documentation for Available Applications.](#)
- [2. Create New cFS Application](#)
 - [2.1 Create New App Items](#)
 - [2.2 Update New App MIDs](#)
 - [2.3 Add New App to Scheduler App](#)
 - [2.4 Add New App to Telemetry Output App](#)
 - [2.5 Add New App to Mission Definitions](#)
 - [2.6 Verify New App is Created Correctly](#)
 - [2.7 Add New App messages to Ground System Tool](#)
 - [2.8 Execute the Ground System Tool](#)
 - [2.9 Commit and Push Changes to the Remote Project Repository](#)
- [3. Add new telemetry to cFS Applications](#)
 - [3.1 Create MID for New Telemetry in New Application](#)
 - [3.2 Add New Telemetry to Telemetry Output Application](#)
 - [3.3 Add New Telemetry Testing to Unit tests.](#)
 - [3.4 Compile and Execute Updated Unit Tests](#)
- [4. Create Developer Integration Test to exercise New Telemetry](#)
 - [4.1 Create Script to send New Telemetry Command](#)
 - [4.2 Test the New Script with FSW](#)
 - [4.3 Add_New_Telemetry_to_Ground_System_Tool](#)
 - [4.4 Execute FSW with New Telemetry Request Command Script and Ground System Tool](#)
- [5. Add cFS App Command to Ground System Tool](#)
 - [5.1 Setup for CHeaderParser.py Execution.](#)
 - [5.2 Add Telemetry Output Housekeeping to Ground System Tool](#)
 - [5.3 Execute CHeaderParser.py to Generate cFS App New Command Files.](#)
 - [5.4 Exercise New cFS App command.](#)
- [6. cFS Test Framework \(CTF\)](#)
 - [6.1 Setup](#)
 - [6.2 Important Files](#)
 - [6.3 Execute a Developer Integration Test using a Terminal](#)
 - [6.4 Create a Developer Integration Test using Scripts](#)
 - [6.5 Run a Developer Integration Test using the CTF GUI](#)
 - [6.6 Create a Developer Integration Test using the CTF GUI \(Under Construction\)](#)
 - [6.7 cFS Test Framework Utilities.](#)
- [Documents](#)

0. Prerequisites

The cFS build process, and thus this guide, depend on having certain software packages installed on your Ubuntu workstation. The following will install you what you need:

```
$ sudo apt install git cmake lcov python3-pip python3-pyqt5 doxygen graphviz
$ pip3 install zmq
```

1. Setup Project Repository

1.1 Complete Setup

1.1.1 Open a Linux (preferably Ubuntu 22.04) terminal for the cFS FSW (we'll call FSW terminal). Increase message queue size.

```
sudo sysctl fs.mqueue.msg_max=2000
```

1.1.2 Using the FSW Terminal, go to the home directory (chosen arbitrarily). Clone the desired spacesuit software repository (we'll call project_repo) and create a new branch (we'll call new_branch).

```
git clone project_repo_https_url (Example: git clone https://gitlab.devops.axiomspace.com/eva/xinfo/ax-xinfo.git)
cd project_repo
git checkout -b new_branch (Note: remove -b if branch already exists in remote repo)
```

1.2 Verify_Setup_is_Correct

1.2.1 Using the FSW terminal, go to the folder project_repo. Complete build preparation. Replace mission with the subsystem specific mission (like axinfo):

```
make distclean
make SIMULATION=native ENABLE_UNIT_TESTS=true MISSIONCONFIG=mission_prep
```

1.2.2 Using the FSW terminal, compile the code:

```
make
```

1.2.3 Using the FSW terminal, compile and execute the existing unit tests (Ignore errors for now):

```
make test
```

1.2.4 Using the FSW terminal, run line coverage on code:

```
make lcov
```

1.2.5 Using the FSW terminal, view the code coverage report by right clicking on link at the bottom and press button "Open Link".

Open Link

1.2.6 Using the FSW terminal, deploy the code to the targets:

```
make install
```

1.2.7 Using the FSW terminal, execute the code (integration level "black-box" execution):

```
cd build/exe/cpul
./core-cpul -R PO
```

Note: Pipe command into tee program like below to capture output on screen and in file log.txt

```
./core-cpul -R PO | tee log.txt
```

Use an editor to examine the output file log.txt. Verify there are no CFE, PSP, OSAL, CI_LAB, TO_LAB, SCH_LAB, or SAMPLE application errors in the output.

1.3 Execute_FSW_with_Ground_System_Tool

1.3.1 Open another Linux terminal (we'll call GS terminal) to execute the Ground System Tool. Go to folder `project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil` and compile Ground System Command Utility:

```
cd tools/cFS-GroundSystem/Subsystems/cmdUtil
make
```

1.3.2 If necessary, using the GS Terminal, go to folder `project_repo/tools/cFS-GroundSystem`. Setup the Ground System tool `cFS-GroundSystem` execution:

```
cd ../../
pip3 install -r requirements.txt
pip3 install -e ~/bitbucket_repo/tools/cFS-GroundSystem
```

1.3.3 Using the GS Terminal, go to folder `project_repo/tools/cFS-GroundSystem` and execute the Ground System tool using:

```
cFS-GroundSystem OR
python3 GroundSystem.py (preferred if changes to cFS-GroundSystem are required)
python3 GroundSystem.py
```

1.3.4 Using the Ground System Tool GUI, open the telemetry GUI by pressing button "Start Telemetry System".

Start Telemetry System

1.3.5 Using the Ground System Tool GUI, open the command GUI by pressing button "Start Command System".

Start Command System

1.3.6 Using the Ground System Tool Command GUI, enable telemetry by pressing button "Enable Tlm".

Enable Tlm

1.3.7 Using the Ground System Tool Command GUI, input IP address in the command dialog box.

127.0.0.1

1.3.8 Using the Ground System Tool, in the command dialog box, send command by pressing button "send".

send

1.3.9 Using the Ground System Tool Telemetry GUI, verify the Ground System Tool receives telemetry from the sample app.

Event Message	0x808	1
ES HK Tlm	0x800	1
EVS HK Tlm	0x801	1
SB HK Tlm	0x803	1
TBL HK Tlm	0x804	1
TIME HK Tlm	0x805	1

1.3.10 Using the Ground System Tool Command GUI, send the Sample No Op command to `cpu1` by pressing button "Sample No-Op"

Sample No-Op

1.3.11 Using the FSW Terminal, in folder `project_repo/build/exe/cpu1`, verify `cpu1` received the Sample No-Op command.

```
EVS Port1 66/1/SAMPLE_APP 3: SAMPLE: NOOP command v1.3.0-rc4+dev21
```

1.3.12 Using the FSW terminal, in `project_repo/build/exe/cpu1`, kill `cpu1`.

```
Ctrl-C
```

1.3.13 Using the GS terminal, in folder `project_repo/tools/cFS-GroundSystem`, kill the Ground System Tool.

```
Ctrl-C
```

1.4 Create Documentation for Available Applications.

1.4.1 Using the FSW Terminal, go to the folder `project_repo`. Generate documentation using one of the following last three commands. Right click on given link and press "Open Link" to view in a web browser.

```
cd ~/project_repo
make doc
make usersguide
make osalguide
```

2. Create New cFS Application

2.1 Create New App Items

2.1.1 Using the FSW terminal, in folder `project_repo/apps/axinfo_apps`, create a new application folder (we'll call `new_app`) using script `create_new_app.sh`. Enter the name of the new application in all lower case (we'll call `new_app`) as input argument. This script creates the new app folder, renames the files, and updates the file contents by replacing instances of `sample` with the new app name.

```
./create_new_app.sh new_app
```

2.2 Update New App MIDs

2.2.1 Using the FSW terminal, in folder `project_repo/apps/subsystem_apps/new_app/fsw/platform_inc`, open file `new_app_msgids.h` using an editor (like `gedit`), change MID numbers to be different than `sample_app`

```
gedit new_app_msgids.h
```

Before:

```
#define NEW_APP_CMD_MID      0x1882
#define NEW_APP_SEND_HK_MID 0x1883
/* V1 Telemetry Message IDs must be 0x08xx */
#define NEW_APP_HK_TLM_MID 0x0883
```

After:

```
#define NEW_APP_CMD_MID      0x18C2
#define NEW_APP_SEND_HK_MID 0x18C3
/* V1 Telemetry Message IDs must be 0x08xx */
#define NEW_APP_HK_TLM_MID 0x08C3
```

2.3 Add New App to Scheduler App

2.3.1 Using FSW terminal, in folder project_repo/apps/sch_lab/, file CMakeLists.txt, add new_app to line below. Save and close the file.

```
foreach(EXT_APP ci_lab to_lab sample_lib new_app)
```

2.3.2 Using FSW terminal, in folder project_repo/apps/sch_lab/fsw/tables, file sch_lab_table.c, add line below after line 29 (generic_app_msgids.h line)

```
#include "new_app_msgids.h"
```

2.3.3 Using FSW terminal, in same file sch_lab_table.c, add line below after line 58

```
{CFE_SB_MSGID_WRAP_VALUE(NEW_APP_SEND_HK_MID), 4, 0},
```

2.4 Add New App to Telemetry Output App

2.4.1 Using an editor, open file project_repo/apps/to_lab/CMakeLists.txt. Add new_app to line below. Save and close the file.

```
foreach(EXT_APP ci_lab to_lab sample_lib new_app)
```

2.4.2 Using an editor, open file project_repo/apps/to_lab/fsw/tables/to_lab_sub.c, add line below after line 39 (sample_app_msgid line)

```
#include "new_app_msgids.h"
```

2.4.3 Using an editor, in same file to_lab_sub.c, add line below after line 53

```
{CFE_SB_MSGID_WRAP_VALUE(NEW_APP_HK_TLM_MID), {0, 0}, 4},
```

2.5 Add New App to Mission Definitions

2.5.1 Using an editor, open file project_repo/mission_defs/targets.cmake. Add new_app to MISSION_GLOBAL_APPLIST .

Before:

```
list(APPEND MISSION_GLOBAL_APPLIST sample_app sample_lib)
```

After:

```
list(APPEND MISSION_GLOBAL_APPLIST new_app sample_app sample_lib)
```

2.5.2 Using an editor, open file project_repo/mission_defs/cpu1_cfe_es_startup.scr. Add new line below after line 3 (sample_app line).

```
CFE_APP, new_app, NEW_APP_Main, NEW_APP, 51, 16384, 0x0, 0;
```

2.6 Verify New App is Created Correctly

2.6.1 Using the FSW terminal, go to folder project_repo. Remove old build using:

```
cd project_repo  
make distclean
```

2.6.2 Repeat section "[Verify_Setup_is_Correct](#)"

2.7 Add New App messages to Ground System Tool

2.7.1 Using an editor, open file project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui/command-pages.txt, add line below after line 31 (Sample App, line)

```
New App, NEW_APP_CMD, 0x18C2, LE, UdpCommands.py, 127.0.0.1, 1234
```

2.7.2 Using an editor, in same file command-pages.txt, add line below after line 43 (Sample App (CPU1) line)

```
New App (CPU1), NEW_APP_CMD, 0x18C2, LE, UdpCommands.py, 127.0.0.1, 1234
```

2.7.3 Using an editor, open file project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui/quick-buttons.txt, add line below after line 33 (SAMPLE line)

```
New App, NEW_APP_CMD, New No-Op, 0, 0x18C2, LE, 127.0.0.1, 1234, NEW_APP_NOOP_CC
```

2.7.4 Using an editor, open file project_repo/tools/cFS-GroundSystem/Subsystems/tlmGUI/telemetry-pages.txt, add line below after line 13 (Sample HK line)

```
New HK Tlm, GenericTelemetry.py, 0x8C3, sample-hk-tlm.txt
```

2.8 Execute the Ground System Tool

2.8.1 Using the GS terminal, repeat section "[Execute FSW with Ground System Tool](#)"

2.8.2 Using the Ground System Tool Command GUI, send the Sample No Op command to cpu1 by pressing button "New No-Op"

New No-Op

2.8.3 Using the FSW Terminal, verify cpu1 received the Sample No-Op command.

```
EVS Port1 66/1/NEW_APP 3: SAMPLE: NOOP command v1.3.0-rc4+dev21
```

2.8.4 Using the FSW Terminal, kill cpu1 execution.

```
Ctrl-C
```

2.8.5 Using the Ground System Tool GUI, kill the Ground System Tool by pressing button close.

```
Ctrl-C
```

2.9 Commit and Push Changes to the Remote Project Repository

2.9.1 Using the FSW Terminal, go to folder project_repo. Commit changes locally

```
git branch
git status
git add space-separated-list-of-unrevised-items-from-above
git status
git commit -m "Jira Story Number: Note"
```

2.9.2 Using the FSW Terminal, push changes to the remote repo in bitbucket/gitlab

```
git push
```

3. Add new telemetry to cFS Applications

3.1 Create MID for New Telemetry in New Application

3.1.1 Using an editor, open file project_repo/apps/subsystem_apps/new_app/fsw/platform_inc/new_app_msgids.h, create telemetry MID for new telemetry message after line 30

Before:

```
#define NEW_APP_HK_TLM_MID 0x08C3
```

After:

```
#define NEW_APP_HK_TLM_MID 0x08C3
#define NEW_APP_NEW_TLM_MID 0x08C4
```

3.1.2 Using an editor, open file project_repo/apps/subsystem_apps/new_app/fsw/src/new_app_msg.h. Create command code and data field size for new telemetry request command after line 32.

Before:

```
#define NEW_APP_PROCESS_CC 2
```

After:

```
#define NEW_APP_PROCESS_CC 2
#define NEW_APP_NEW_TLM_CC 3
/*
** New App Message Sizes
*/
#define NEW_APP_NEW_TLM_SIZE_BYTES 16
```

3.1.3 Using an editor, in the same file, create a structure for the new telemetry message based on data dictionary after line 73.

Before:

```
} NEW_APP_HkTlm_t;
```

After:

```
} NEW_APP_HkTlm_t;
typedef struct
{
CFE_MSG_TelemetryHeader_t TelemetryHeader; /**< \brief Telemetry header */
uint8 ucCommandCounter;           /**< \brief Command Counter */
uint8 ucCommandErrorCounter;      /**< \brief Command Error Counter */
uint8 ucDataField[NEW_APP_NEW_TLM_SIZE_BYTES]; /**< \brief Data Field */
/* . . .< add variables based on data dictionary */
} NEW_APP_NewTlm_t;
```

3.1.4 Using an editor, open file project_repo/apps/subsystem_apps/new_app/fsw/src/new_app.h. Add New App global member variable for new telemetry message after line 71

```
cd apps/new_app/fsw/src
```

Before:

```
New_APP_HkTlm_t HkTlm;
```

After:

```
New_APP_HkTlm_t HkTlm;
/*
** New telemetry packet...
*/
New_APP_NewTlm_t NewTlm;
```

3.1.5 Using an editor, in the same file, add New App new telemetry message function prototypes after line 103.

Before:

```
int32 NEW_APP_ReportHousekeeping(const CFE_MSG_CommandHeader_t *Msg);
```

After:

```
int32 NEW_APP_ReportHousekeeping(const CFE_MSG_CommandHeader_t *Msg);
int32 NEW_APP_ProcessNewTelemetry(const NEW_APP_NewTlm_t *Msg);
```


3.1.6 Using an editor, open file `project_repo/apps/subsystem_apps/new_app/fsw/src/new_app.c`, in function `New_App_Init`, initialize new telemetry global variable after line 146.

Before:

```
CFE_MSG_Init(CFE_MSG_PTR(NEW_APP_Data.HkTlm.TelemetryHeader), CFE_SB_ValueToMsgId(NEW_APP_HK_TLM_MID),
sizeof(NEW_APP_Data.HkTlm));
```

After:

```
CFE_MSG_Init(CFE_MSG_PTR(NEW_APP_Data.HkTlm.TelemetryHeader), CFE_SB_ValueToMsgId(NEW_APP_HK_TLM_MID),
sizeof(NEW_APP_Data.HkTlm));
/*
** Initialize new telemetry packet
*/
CFE_MSG_Init(CFE_MSG_PTR(NEW_APP_Data.NewTlm.TelemetryHeader), CFE_SB_ValueToMsgId(NEW_APP_NEW_TLM_MID),
sizeof(NEW_APP_Data.NewTlm));
```

3.1.7 Using an editor, in the same file, in function `NEW_APP_ProcessGroundCommand`, add case condition for new telemetry message command code to call new telemetry process function.

Before:

```
case NEW_APP_PROCESS_CC:
if (NEW_APP_VerifyCmdLength(&SBBufPtr->Msg, sizeof(NEW_APP_ProcessCmd_t)))
{
NEW_APP_Process((NEW_APP_ProcessCmd_t *)SBBufPtr);
}
break;
```

After:

```
case NEW_APP_PROCESS_CC:
if (NEW_APP_VerifyCmdLength(&SBBufPtr->Msg, sizeof(NEW_APP_ProcessCmd_t)))
{
NEW_APP_Process((NEW_APP_ProcessCmd_t *)SBBufPtr);
}
break;

case NEW_APP_NEW_TLM_CC:
if (NEW_APP_VerifyCmdLength(&SBBufPtr->Msg, sizeof(NEW_APP_NewTlm_t)))
{
NEW_APP_ProcessNewTelemetry((NEW_APP_NewTlm_t *)SBBufPtr);
}
break;
```

3.1.8 Using an editor, in the same file, create new function implementation `NEW_APP_ProcessNewTelemetry`, after line 322.

Before:

```
} /* End of NEW_APP_ReportHousekeeping() */
```

After:

```

} /* End of NEW_APP_ReportHousekeeping() */
/* **** */
/* Name: NEW_APP_ProcessNewTelemetry */
/* */
/* Purpose: */
/* purpose*/
/* **** */
int32 NEW_APP_ProcessNewTelemetry(const NEW_APP_NewTlm_t *Msg)
{
/*    . . .add functionality */
/*
** Send new telemetry to telemetry output application
*/
CFE_SB_TransmitMsg(CFE_MSG_PTR(NEW_APP_Data.NewTlm.TelemetryHeader), true);
return CFE_SUCCESS;
} /* End of ProcessNewTelemetry () */

```

3.1.9 Using an editor, open file `project_repo/apps/to_lab/fsw/tables/to_lab_sub.c`. Add new telemetry line to TO_LAB Subscription after line 59.

Before:

```
{CFE_SB_MSGID_WRAP_VALUE(NEW_APP_HK_TLM_MID), {0, 0}, 4},
```

After:

```
{CFE_SB_MSGID_WRAP_VALUE(NEW_APP_HK_TLM_MID), {0, 0}, 4},
{CFE_SB_MSGID_WRAP_VALUE(NEW_APP_NEW_TLM_MID), {0, 0}, 4},
```

3.1.10 Complete sections “[Verify Setup is correct](#)” and “[Execute FSW with Ground System Tool](#)” to verify new code passes compilation and FSW execution with Ground System Tool.

3.2 Add New Telemetry to Telemetry Output Application

3.2.1 Using an editor, open file `project_repo/apps/to_lab/fsw/tables/to_lab_sub.c` . Below line 66, add new telemetry line below to telemetry output subscription.

```
{CFE_SB_MSGID_WRAP_VALUE(NEW_APP_NEW_TLM_MID), {0, 0}, 4},
```

3.3 Add New Telemetry Testing to Unit tests.

3.3.1 Using an editor, open file `project_repo/apps/subsystem_apps/new_app/unit-test/coveragetest/coveragetest_new_app.c`. In function `Test_NEW_APP_ProcessGroundCommand` add new telemetry variable to `TestMsg` structure after line 338.

Before:

```
NEW_APP_ProcessCmd_t Process;
```

After:

```
NEW_APP_ProcessCmd_t Process;
NEW_APP_NewTlm_t NewTlm;
```

3.3.2 Using an editor, in the same file and function, add test for new telemetry command code after line 385.

Before:

```
FcnCode = NEW_APP_PROCESS_CC;
Size = sizeof(TestMsg.Process);
UT_SetDefaultReturnValue(UT_KEY(CFE_TBL_GetAddress), CFE_TBL_ERR_UNREGISTERED);
UT_SetDataBuffer(UT_KEY(CFE_MSG_GetFcnCode), &FcnCode, sizeof(FcnCode), false);
UT_SetDataBuffer(UT_KEY(CFE_MSG_GetSize), &Size, sizeof(Size), false);
NEW_APP_ProcessGroundCommand(&TestMsg.SBBuf);
```

After:

```
FcnCode = NEW_APP_PROCESS_CC;
Size = sizeof(TestMsg.Process);
UT_SetDefaultReturnValue(UT_KEY(CFE_TBL_GetAddress), CFE_TBL_ERR_UNREGISTERED);
UT_SetDataBuffer(UT_KEY(CFE_MSG_GetFcnCode), &FcnCode, sizeof(FcnCode), false);
UT_SetDataBuffer(UT_KEY(CFE_MSG_GetSize), &Size, sizeof(Size), false);
NEW_APP_ProcessGroundCommand(&TestMsg.SBBuf);

FcnCode = NEW_APP_NEW_TLM_CC;
Size = sizeof(TestMsg.NewTlm);
UT_SetDefaultReturnValue(UT_KEY(CFE_TBL_GetAddress), CFE_TBL_ERR_UNREGISTERED);
UT_SetDataBuffer(UT_KEY(CFE_MSG_GetFcnCode), &FcnCode, sizeof(FcnCode), false);
UT_SetDataBuffer(UT_KEY(CFE_MSG_GetSize), &Size, sizeof(Size), false);
NEW_APP_ProcessGroundCommand(&TestMsg.SBBuf);
```

3.3.3 Using an editor, in the same file, add a new test function below for new telemetry processing after implementation of function Test_NEW_APP_ProcessGroundCommand(void) {} after line 417.

```
void Test_NEW_APP_ProcessNewTelemetry(void)
{
    /*
    * Test Case For:
    * void NEW_APP_ProcessNewTelemetry( const NEW_APP_NewTlm *Msg )
    */
    NEW_APP_NewTlm_t Msg;

    /* Set up to capture send message address */
    UT_SetDataBuffer(UT_KEY(CFE_SB_TransmitMsg), &Msg, sizeof(Msg), false);
    /* Call unit under test */
    NEW_APP_ProcessNewTelemetry(&Msg);
    /* Confirm transmit msg */
    UtAssert_STUB_COUNT(CFE_SB_TransmitMsg, 1);
    // Add more unit tests here
}
```

3.3.4 Using an editor, in the same file, add the new unit test to the unit test list after line 646.

Before:

```
ADD_TEST(NEW_APP_ReportHousekeeping);
```

After:

```
ADD_TEST(NEW_APP_ReportHousekeeping);
ADD_TEST(NEW_APP_ProcessNewTelemetry);
```

3.4 Compile and Execute Updated Unit Tests

3.4.1 Using the FSW Terminal, go to folder `project_repo/build/native/default_cpu1/apps/subsystem_apps/new_app/unit-test`. Compile the updated `new_app` code with the updated unit tests. To debug compilation, update the code and unit tests and complete this step.

```
cd build/native/default_cpu1/apps/subsystem_apps/new_app/unit-test
make
```

3.4.2 Using the FSW Terminal, in the same folder, run the updated `new_app` code with the updated unit tests. To debug execution, update the code and unit tests and repeat the previous step and this step.

```
./coverage-new_app-ALL-testrunner
```

4. Create Developer Integration Test to exercise New Telemetry

4.1 Create Script to send New Telemetry Command

4.1.1 Using the new terminal (We'll call `cmdUtil` Terminal), go to folder `project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil`. Copy `readmet.txt` to create a new script `new_app_new_telemetry_req_cmd.sh`.

```
cd ~/project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil
cp readme.txt new_app_new_telemetry_req_cmd.sh
chmod +x new_app_new_telemetry_req_cmd.sh
```

4.1.2 Using an editor, open the new file from above. Comment out the existing lines for reference, and add below to the bottom of the file correcting the Packet ID, Command Code, and second string number of bytes to fit the new telemetry.

```
echo "send new telemetry request command"
./cmdUtil --host=localhost --port=1234 --endian=LE --pktid=0x18E2 --cmdcode=3 --string="8:" --string="18:"
sleep 1
```

Note: Verify `-string="20:"` size with: `~/project_repo/apps/new_app/fsw/src/*msg.h` total command variable bytes – excluding header.

4.2 Test the New Script with FSW

4.2.1 Using the FSW Terminal, go to folder `~/project_repo/build/exe/cpu1`. Execute the FSW `cpu1`.

```
cd ~/project_repo/build/exe/cpu1
./core-cpu1 -R PO
```

4.2.2 Using the `cmdUtil` Terminal, go to folder `~/project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil`. Execute the new telemetry request command script.

```
cd ~/project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil
./new_app_new_telemetry_req_cmd.sh
```

4.2.3 Using the FSW Terminal, for `cpu1` execution, verify there are no errors like below:

```
EVS Port1 66/1/IO_APP 6: Invalid Msg length: ID = 0x18C2, CC = 4, Len = 718, Expected = 720
```

4.3 Add_New_Telemetry_to_Ground_System_Tool

4.3.1 Using the GS Terminal, go to folder ~/project_repo/tools/cFS-GroundSystem/Subsystems/tlmGUI. Copy file sample-hk-tlm.txt to create a new file new_app_new_telemetry-tlm.txt for the new telemetry.

```
cp generic-app-hk-tlm.txt new_app_new_telemetry-tlm.txt
```

4.3.2 Using an editor, open the new file new_app_new_telemetry-tlm.txt. Edit the file based on the new telemetry structure in file ~/project_repo/apps/subsystem_apps/new_app/fsw/src/new_app_msg.h.

Example:

Contents of file new_app_msg.h:

```
#define NEW_APP_NEW_TLM_SIZE_BYTES 16
typedef struct
{
    CFE_MSG_TelemetryHeader_t TelemetryHeader; /**< \brief Telemetry header */
    uint8 ucCommandCounter; /**< \brief Command Counter */
    uint8 ucCommandErrorCounter; /**< \brief Command Error Counter */
    uint8 ucDataField[NEW_APP_NEW_TLM_SIZE_BYTES]; /**< \brief Data Field */
} NEW_APP_NewTlm_t;
```

Contents of file new_app_new_telemetry-tlm.txt:

ucCommandCounter,	12,	1,	B,	Dec,	NULL,	NULL,	NULL,	NULL
ucCommandErrorCounter,	13,	1,	B,	Dec,	NULL,	NULL,	NULL,	NULL
ucDataField,	14,	16,	B,	Dec,	NULL,	NULL,	NULL,	NULL

4.3.3 Using an editor, open file project_repo/tools/cFS-GroundSystem/Subsystems/tlmGUI/telemetry -pages.txt. Add the line below for the new telemetry replacing 3rd item with MID NEW_APP_NEW_TLM_MID value from file ~/project_repo/apps/subsystem_apps/new_app/fsw/platform_inc/new_app_msgids.h

```
New app New Telemetry Tlm, GenericTelemetry.py, 0x08C4, new_app_new_telemetry-tlm.txt
```

4.4 Execute FSW with New Telemetry Request Command Script and Ground System Tool

4.4.1 Using the FSW Terminal, verify cpu1 is still executing with no errors.

4.4.2 Using the Ground System Terminal, go to folder ~/project_repo/tools/cFS-GroundSystem. Complete section “[Execute FSW with Ground System Tool](#)” steps 1.3.3. to 1.3. 10.

4.4.3 Using the cmdUtil Terminal, go to folder ~/project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil. Execute the new telemetry request command script.

```
cd ~/project_repo/tools/cFS-GroundSystem/Subsystems/cmdUtil
./new_telemetry_req_cmd.sh
```

Note: Pipe command into tee program like below to capture output on screen and in file log.txt

```
./new_telemetry_req_cmd.sh | tee log.txt
```

4.4.4 Using the FSW Terminal, for cpu1 execution, verify there are no errors like below:

```
EVS Port1 66/1/IO_APP 6: Invalid Msg length: ID = 0x18C2, CC = 4, Len = 718, Expected = 720
```

4.4.5 Using the Ground System Tool Telemetry GUI, verify the new telemetry counter increments.

```
New App New Tlm 0x8c4 1
```

5. Add cFS App Command to Ground System Tool

(The Telemetry Output App Add Packet command and Telemetry Output Housekeeping Telemetry will be used as an example)

5.1 Setup for CHeaderParser.py Execution.

5.1.1 Open a new terminal (we'll call cmdGui terminal), and go to folder project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui. Copy the to_lab msg file to the cFS Ground System Command GUI folder

```
cd project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui
cp ../../../../apps/to_lab/fsw/src/to_lab_msg.h to_lab_msg_for_grnd.h
```

5.1.2 Using an editor open file ~/project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui/to_lab_msg_for_grnd.h. Around line 106, update the Add Packet command definition so that it only contains primitive variables except for the command header.

Before:

```
typedef struct
{
    CFE_MSG_CommandHeader_t    CmdHeader; /**< \brief Command header */
    TO_LAB_AddPacket_Payload_t Payload;    /**< \brief Command payload */
} TO_LAB_AddPacketCmd_t;
```

After:

```
typedef struct
{
    CFE_MSG_CommandHeader_t    CmdHeader; /**< \brief Command header */
    uint32 Stream;
    uint16 PktSize;
    uint16 Flags;
    uint8  BufLimit;
} TO_LAB_AddPacketCmd_t;
```

5.1.3 Using an editor, open file project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui/CHeaderParser-hdr-paths.txt. Add a new line at the bottom of the file for the msg file. Comment out all other lines in the file.

Add Line:

```
./to_lab_msg_for_grnd.h
```

5.2 Add Telemetry Output Housekeeping to Ground System Tool

5.2.1 Complete section “[Add_New_Telemetry_to_Ground_System_Tool](#)” to add new telemetry to Ground System Tool.

5.3 Execute CHeaderParser.py to Generate cFS App New Command Files.

5.3.1 Using cmdGui Terminal), go to folder project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui. Execute CommandParser.py and walk through steps to create the new command.

```
cd project_repo/tools/cFS-GroundSystem/Subsystems/cmdGui
python3 CHeaderParser.py
...
What would you like the command file to be saved as? TO_LAB_APP_CMD
...
Do any commands in TO_LAB_APP_CMD require parameters? (yes/no): yes
Enter a value from the list above or -1 to exit: 3
Enter a value from the list above or -1 to exit: 7
Enter the line of the parameter from the above print-out (-1 to stop): 2
Please enter parameter description: note
Enter the line of the parameter from the above print-out (-1 to stop): 2
Please enter parameter description: note
Enter the line of the parameter from the above print-out (-1 to stop): 2
Please enter parameter description: note
Enter the line of the parameter from the above print-out (-1 to stop): 2
Please enter parameter description: note
Enter the line of the parameter from the above print-out (-1 to stop): -1
Enter a value from the list above or -1 to exit: -1
Exiting.
Thank you for using CHeaderParser.
The following commands have been added with parameters:
['TO_ADD_PKT_CC']
```

5.4 Exercise New cFS App command.

5.4.1 Complete section “[Execute_FSW_with_Ground_System_Tool](#)” to setup new command execution.

5.4.2 Using the Ground System Tool Command GUI, select Telemetry Output button "Display Page".

Display Page

5.4.3 Using the Ground System Tool Telemetry Output Display Page, press TO_REMOVE_PKT_CC (Add Packet) button "Send".

Send

5.4.4 Using the new dialog box, in TO_REMOVE_PKT_CC Dialog box, enter values below for parameter inputs (leave description empty) and press "Send".

Stream: 2176

Send

5.4.5 Using the Ground System Tool Telemetry GUI, verify the Telemetry Output housekeeping Counter stops incrementing.

TO_LAB_APP HK Tlm 0x0880 #

5.4.6 Using the Ground System Tool Telemetry Output Display Page, press TO_ADD_PKT_CC (Add Packet) button "Send".

Send

5.4.7 Using the new dialog box, in TO_ADD_PKT_CC Dialog box, enter values below for parameter inputs (leave description empty) and press "Send".

```
Stream: 2176  
PktSize: 0  
Flags: 0  
BufLimit: 4
```

Send

5.4.8 Using the Ground System Tool Telemetry GUI, verify the Telemetry Output housekeeping Counter increments.

```
TO_LAB APP HK Tlm 0x0880 #
```

6. cFS Test Framework (CTF)

The cFS Test Framework is used to create and run cFS SW developer integration tests and requirements verification tests.

6.1 Setup

6.1.1 Read CTF User's Guide "CTF_SUG.pdf" all sections.

6.1.2 Open a new Linux terminal (we'll call CTF Terminal), go to folder project_repo/tools/CTF. Setup Anaconda Environment (complete once)

```
cd ~/project_repo/tools/CTF
source setup_ctf_env.sh
```

6.1.3 Using the CTF Terminal, activate the Anaconda Environment.

```
cd ~/project_repo/tools/ctf
source activate_ctf_env.sh
```

6.2 Important Files

6.2.1 CTF configuration files *config.ini in ~/project_repo/tools/ctf_tests/configs. These files are inputs to CTF that contain CTF configuration settings.

Example: default_config.ini

```
#####
# Note - This config file provides an example configuration to run CTF as a CFS tool inside a CFS workspace.
#       Please ensure that the appropriate fields are set below, specifically for the [cfs] section to match
your
#       CFS workspace layout.
#####

#####
# CTF Core Configuration
#####

[core]

# Global verification timeout for any "verify_required" CTF instructions.
# It can be overridden by file-scope attribute or instruction-scope attribute "verify_timeout".
# Unit: Seconds if using generic system time manager)
ctf_verification_timeout = 5.0

# Global polling period for any "verify_required" CTF instruction.
# Defines how often to run verification commands until verification either
# passes or timeout
# Unit: Seconds if using generic system time manager)
ctf_verification_poll_period = 0.5

# Reset plugins between scripts? This is useful if
# scripts assume a fresh state of CFS/Trick_CFS
# If set to false, plugins will not shutdown/re-initialize
# between scripts
reset_plugins_between_scripts = true

# How long to wait between test scripts
delay_between_scripts = 3.0

# End test on fail?
end_test_on_fail = true

# Paths of additional plugins to be loaded/used by CTF. Comma-separated.
# All plugins within that directory will be loaded unless
# disabled explicitly in `disabled_plugins`.
# Note - The additional plugin directory requires a unique name
#       in order to not shadow any CTF top-level directories.
#       Do not name the directory 'plugins', 'lib' or other folder
#       names within the CTF repo.
additional_plugins_path =

# Disabled plugins (directory name of plugin). Comma-separated
disabled_plugins =

# Ignored plugin test instructions. Comma-separated
```

```

ignored_instructions =

#####
# CTF logging
#####

[logging]

# Output directory for CTF scripts. Relative to directory CTF is launched from.
results_output_dir = ./CTF_Results

# Filename for CTF Log
ctf_log_file = CTF_Log_File.log

# Generate a json version of the regression_results_summary (true)
# in addition to the text file version
json_results = True

# What level of logging?
# ERROR : only show error logs - very minimal output
# INFO : only show info, warning, error, and critical logs
# DEBUG: show all logs!
log_level = INFO

#####
# ccsds options
#####
[ccsds]
# Header Info Included in CCSDS Exports?
CCSDS_header_info_included = false

####
# This python module contains the definitions of header types to be exposed in the CCSDS plugin.
# These classes will be used by the CFS plugin to construct command and telemetry packets with
# the appropriate header formats.
####
CCSDS_header_path = ./plugins/ccsds_plugin/cfe/ccsds_v2/ccsds_v2.py

#####
# Base settings for cfs
#####

[cfs]

# CFS workspace directory. All cfs paths will be relative to workspace dir.
#####
#
# Note - The CFS Workspace Directory is the CFS project root directory
# which the remaining configuration fields will utilize.
#
# Note - Other sections with the same fields can be defined as their own CFS targets to be registered.
# example: [my_custom_cfs_target]
# cfs_protocol = local # REQUIRED for each target
# other fields if needed
#####

workspace_dir = ~/sample_cfs_workspace

# cfs protocol setting either:
# local (local host)
# ssh (ssh to host)
# sp0 (sp0 host)
cfs_protocol = local

# Build the CFS project?
build_cfs = true

# Build directory for the CFS project
cfs_build_dir = ${cfs:workspace_dir}

```

```

# Build command to run
cfs_build_cmd = make; make install

# Run directory for the CFS project
cfs_run_dir = ${cfs:workspace_dir}/build/exe/lx1

# Executable to run within the cfs_run_dir
cfs_exe = core-lx1

# Include CFS UDP port in arg (-p portNum)?
cfs_port_arg = False

# Additional arguments to CFS. If it does not include '-RPR', the ram drive will be removed before starting cFS
instance.
# Removing ram drive only applies on Linux targets.
cfs_run_args =

# The ram drive path to be deleted. It is only needed, when '-RPR' argument is not included in cfs_run_args.
# It only applies on Linux targets.
cfs_ram_drive_path = /dev/shm/osal:RAM

# File name where CFS output will be saved
# The target name will be prepended to ensure unique files
cfs_output_file = cfs_stdout.txt

# Automatically remove continuous checks once they fail
# Not doing so may flood the output with error messages
remove_continuous_on_fail = True

# CCSDS Data Directory
####
# This directory contains the command and telemetry definitions in JSON format.
# These definitions are used for CTF to command/receive telemetry, as well as
# used by the editor for auto-suggestion features.
####

CCSDS_data_dir = ${cfs:workspace_dir}/ccdd/json

# Log CCSDS Import Process (Logs all messages parsed from CCSDS Data Directory)
log_ccsds_imports = true

# Name of the target in CCSDS data files
CCSDS_target = set1

# What endianness is the target machine
endianess_of_target = little

# Output directory for CFS EVS
evs_log_file = evs_event_msgs.log

# Run in debug mode using GDB?
cfs_debug = false

# Log every telemetry packet received?
telemetry_debug = false

# Run in a seperate terminal window as CTF? Note - Disable for CI purposes
cfs_run_in_xterm = True

# IP address of the target system
cfs_target_ip = 127.0.0.1

# CI commanding port
cmd_udp_port = 5010

# Set tlm_udp_port to 0 if you want the os to
# choose the port.
# If you want to manually set tlm_udp_port than set it equal to
# the port you want to use
tlm_udp_port = 1235

```

```

# Do you want to use TO or DIAG. Needs to be the exact name of the class to be used
# tlm_app_choice = DiagApi
tlm_app_choice = ToApi

# What CCSDS version
ccsds_ver = 2

# How long to look back in the evs messages to validate
# Note - Setting this value to 0 means CheckEvent packet must
#         be received *while* polling that instruction.
#         Setting this value to X will allow CTF to validate
#         events received with the past X time-units, as well
#         as new events packets
evs_messages_clear_after_time = 5

# Name of MID to collect cfe EVS long messages
evs_event_mid_name = CFE_EVS_LONG_EVENT_MSG_MID

# Name of MID to collect cfe EVS short messages
evs_short_event_mid_name = CFE_EVS_SHORT_EVENT_MSG_MID

#####
# Base settings for local SSH
#####
[local_ssh]
# cfs protocol setting either:
# local    (local host)
# ssh      (ssh to host)
# sp0      (sp0 host)
cfs_protocol = ssh

# Destination IP/hostname
destination = localhost

# Build directory for the CFS project
cfs_build_dir = ${cfs:workspace_dir}

# Run directory for the CFS project
cfs_run_dir = ${cfs:workspace_dir}/build/exe/lx1

# Log every telemetry packet received?
telemetry_debug = false

[ssh]

# Command timeout for the execution plugin
command_timeout = 60

# Print stdout while command is running?
print_stdout = False

# Log stdout when command complete?
log_stdout = True

#####
# Test Variable Configuration
#####
[test_variable]
# Supported data types: int, boolean, float, string
# Variables defined in this section can be used in the same way as 'SetUserVariable' and related instructions.
variable_1 = 10

variable_2 = false

variable_3 = 5.0

variable_4 = "abc"

```

6.2.2 cFS App CCDD JSON files in ~/project_repo/tools/ctf_tests/ccdd/json. These files are inputs to CTF that contain cFS App Message definitions.

Example: auto_ci_CMD.json

```
{
  "cmd_mid_name": "CI_CMD_MID",
  "cmd_description": "",
  "cmd_codes": [
    {
      "cc_name": "CI_NOOP_CC",
      "cc_value": "0",
      "cc_description": "_NOOP_CC value is always 0",
      "cc_data_type": "CI_NoArgCmd_t",
      "cc_parameters": []
    },
    {
      "cc_name": "CI_RESET_CC",
      "cc_value": "1",
      "cc_description": "_RESET_CC value is always 1",
      "cc_data_type": "CI_NoArgCmd_t",
      "cc_parameters": []
    },
    {
      "cc_name": "CI_ENABLE_TO_CC",
      "cc_value": "2",
      "cc_description": "",
      "cc_data_type": "TO_EnableOutputCmd_t",
      "cc_parameters": [
        {
          "name": "cDestIp",
          "description": "",
          "array_size": "16",
          "data_type": "char",
          "bit_length": "0",
          "parameters": []
        },
        {
          "name": "usDestPort",
          "description": "",
          "array_size": "0",
          "data_type": "uint16",
          "bit_length": "0",
          "parameters": []
        },
        {
          "name": "usRouteMask",
          "description": "Route mask to enable",
          "array_size": "0",
          "data_type": "uint16",
          "bit_length": "0",
          "parameters": []
        },
        {
          "name": "iFileDesc",
          "description": "File descriptor of port to use",
          "array_size": "0",
          "data_type": "int32",
          "bit_length": "0",
          "parameters": []
        }
      ]
    }
  ]
}
```

6.2.3 cFS App Function JSON Files in ~/project_repo/tools/ctf_tests/mission_tests. These scripts contain cFS App functions used by CTF Tests.

Example: CiFunctions.json

```
{
  "test_script_number": "Nominal-CI-Commands",
  "test_script_name": "CiFunctions.json",
  "requirements": {
    "MyRequirement": "N/A"
  },
  "description": "Nominal CI Functions",
  "owner": "CTF",
  "test_setup": "",
  "ctf_options": {
    "verify_timeout": 4
  },
  "import": {},
  "functions": {
    "SendCheckCiNoopCmd": {
      "description": "Send and check CI_NOOP_CC",
      "varlist": [
        "expectedCmdCnt",
        "expectedErrCnt"
      ],
      "instructions": [
        {
          "instruction": "SendCfsCommand",
          "data": {
            "target": "",
            "mid": "CI_CMD_MID",
            "cc": "CI_NOOP_CC",
            "args": {}
          },
          "wait": 0
        },
        {
          "instruction": "CheckTlmValue",
          "data": {
            "target": "",
            "mid": "CI_HK_TLM_MID",
            "args": [
              {
                "variable": "usCmdCnt",
                "value": [
                  "expectedCmdCnt"
                ],
                "compare": "=="
              },
              {
                "variable": "usCmdErrCnt",
                "value": [
                  "expectedErrCnt"
                ],
                "compare": "=="
              }
            ]
          },
          "wait": 0
        }
      ]
    },
    "SendCheckCiResetCmd": {
      "description": "Send and check CI_RESET_COUNTER_CC",
      "varlist": [
        "expectedCmdCnt",
        "expectedErrCnt"
      ],
      "instructions": [
        {
          "instruction": "SendCfsCommand",
          "data": {
            "target": "",
```



```

        },
        "wait": 0
    },
    {
        "function": "SendCheckCiLabNoopCmd",
        "wait": 2,
        "params": {
            "expectedCmdErrCnt": 0,
            "expectedCmdCnt": 1,
            "expectedSktConn": 1,
            "expectedIngPkt": 1,
            "expectedIngErr": 0
        }
    },
    {
        {
            "instruction": "ShutdownCfs",
            "data": {
                "target": ""
            },
            "wait": 1
        }
    }
]
}

```

6.3 Execute a Developer Integration Test using a Terminal

6.3.1 Using an editor, open file `project_repo/tools/ctf_tests/mission_tests/mission_config.ini`. Update this file by changing workspace path to `project_repo`.

Before:

```
workspace_dir = ~/cfs
```

After:

```
workspace_dir = ~/project_repo
```

6.3.2 If necessary, using the FSW Terminal, go to folder `project_repo`. Compile the FSW.

```
cd ~/project_repo make
make install
```

6.3.3 Using the CTF Terminal, go to folder `project_repo/tools/CTF`. Execute CTF using the updated mission configuration file and an existing test.

```
cd ~/project_repo/tools/CTF
source activate_ctf_env.sh
./ctf --config_file ../ctf_tests/mission_tests/mission_config.ini ../ctf_tests/mission_tests/CiLabFunctionTests.
json
ls CTF_Results
```

Using an editor, examine test results in folder `project_repo/tools/ctf_tests/CTF_Results`.

6.4 Create a Developer Integration Test using Scripts

6.4.1 Using an editor, open below files to use as a reference for below steps.

1. `project_repo/apps/subsystem_apps/new_app/app/fsw/platform_ic/app_msgids.h`
2. `project_repo/apps/subsystem_apps/new_app/app/fsw/src/new_app_msg.h`

6.4.2 Using the CTF Terminal, go to folder `project_repo/tools/ctf_tests/mission_tests`. Create initial CTF Files for cFS App using Automation Bash Script `AddAppCtfFiles.sh`

```
cd ~/project_repo/tools/ctf_tests/mission_tests
./AddAppCtfFiles.sh app
```

6.4.3 Using an editor, verify the following files were created/updated properly.

- | | |
|--|--|
| 1. <code>~/project_repo/tools/ctf_tests/ccdd/json/cFS/auto_cfs_grp1_MIDs.json</code> . | Verify NEW_APP_MIDs were added to list set 1. |
| 2. <code>~/project_repo/tools/ctf_tests/ccdd/json/auto_new_app_CMD.json</code> . | Verify No Op and Reset Counter commands were created. |
| 3. <code>~/project_repo/tools/ctf_tests/ccdd/json/auto_new_lab_hk_TLM.json</code> .
created. | Verify the Housekeeping Telemetry Data Structure Items were created. |
| 4. <code>~/project_repo/tools/ctf_tests/mission_tests/NEW_APP_Functions.json</code> .
were created. | Verify functions <code>SendCheckNooCmd</code> and <code>SendCheckCiResetCmd</code> were created. |
| 5. <code>~/project_repo/tools/ctf_tests/mission_tests/NEW_APP_FunctionTests.json</code> . | Verify tests NEW_APP-Function-Test-1 was created. |

6.4.4 Using the CTF Terminal, go to folder `project_repo/tools/CTF`. Test the new cFS App CTF Functional Test.

```
cd ~/project_repo/tools/CTF
source activate_ctf_env.sh
ctf --config_file ../ctf_tests/mission_tests/mission_config.ini ../ctf_testsmission_tests/NEW_APP_FunctionTests.json
```

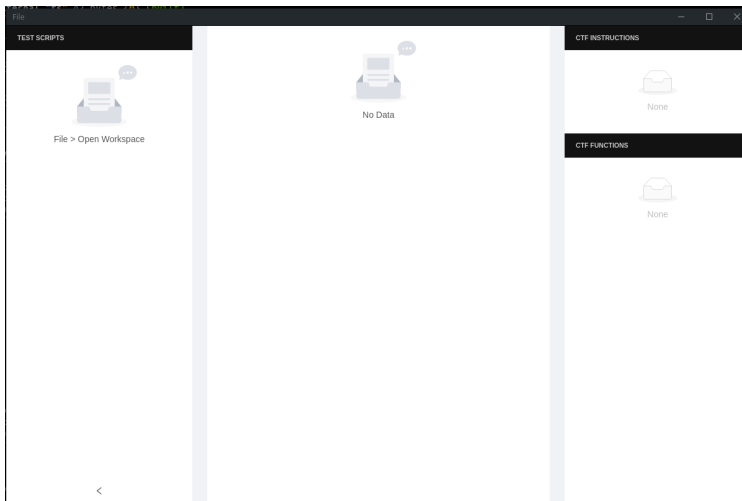
Using an editor, examine test results in folder `project_repo/tools/ctf_tests/CTF_Results`.

6.5 Run a Developer Integration Test using the CTF GUI

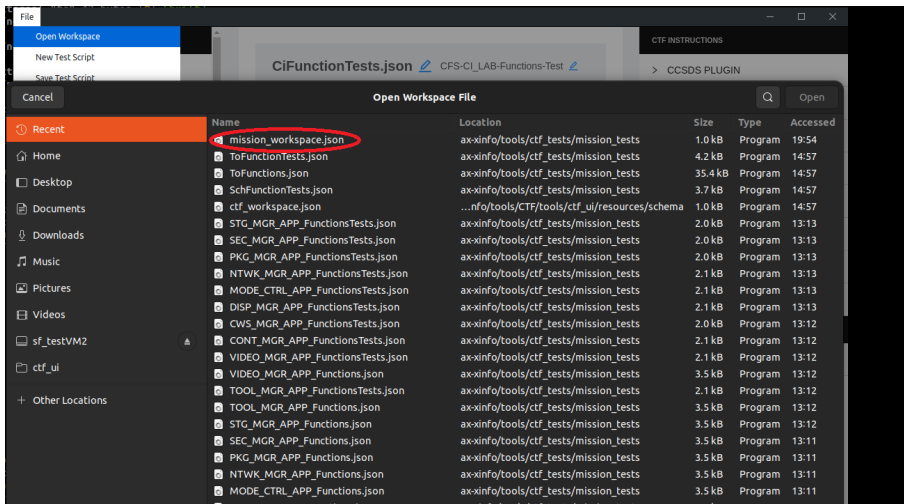
6.5.1 Using the CTF Terminal, go to folder `project_repo/tools/CTF`. Activate CTF environment and execute CTF GUI.

```
cd ~/project_repo/tools/CTF
source activate_ctf_env.sh
./run_editor.sh
```

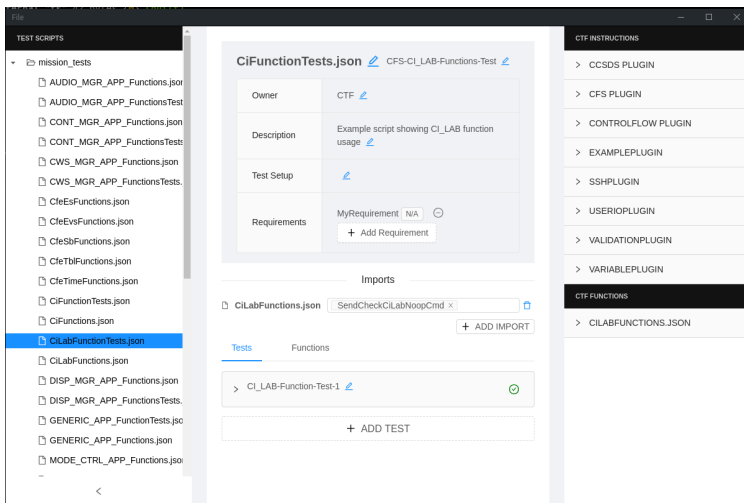
The CTF GUI will open like below.



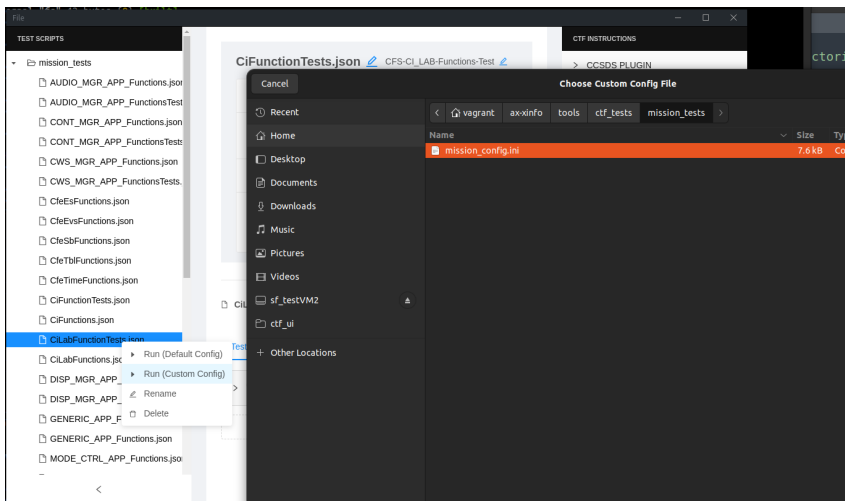
6.5.2 Using the CTF GUI, Select "File" Menu option in the upper left. Select "Open Workspace". Select workspace file `~/project_repo/ax-xinfo/tools/ctf_tests/mission_tests/mission_workspace.com`.



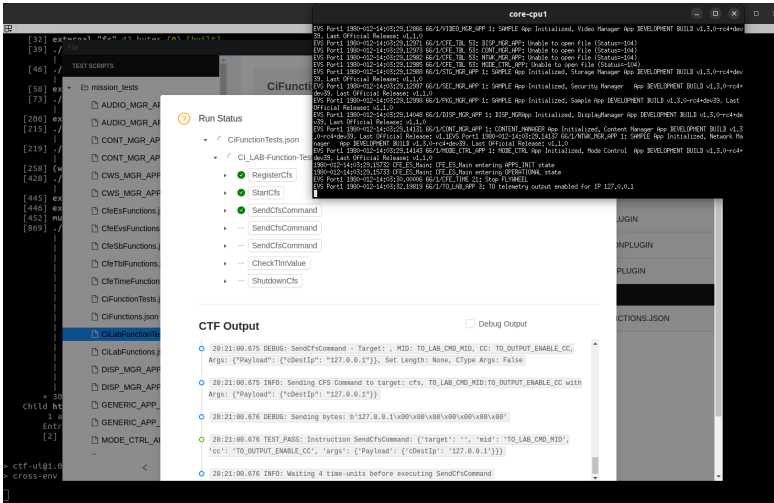
6.5.3 Using the CTF GUI, expand "mission_tests" in the left panel. Select "CiLabFunctionsTests.json" to examine its contents in the middle panel.



6.5.4 Using the CTF GUI, right-click on "CiLabFunctionsTests.json". Select "Run (Custom config)". Select config file ~/project_repo/ax-xinfo/tools/ctf_tests/mission_tests/mission_config.ini to run the CTF test CiLabFunctionsTests.json



The CTF GUI will open a test status dialog box, run the test, and log the results in ~/project_repo/ax-xinfo/tools/CTF/CTF_Results. Click "Done" to close the dialog box.



6.6 Create a Developer Integration Test using the CTF GUI (Under Construction)

6.6.1 Using the CTF Terminal, go to folder project_repo/tools/CTF. Activate CTF environment and execute CTF GUI.

6.7 cFS Test Framework Utilities.


6.7.1 Using CTF Terminal, go to folder project_repo/tools/CTF. Execute CTF utility script "run_test.sh" with options "sca", "utc", "ft", or "vv" to run statistical code analysis, unit tests with code coverage, functional tests, and requirements verification tests respectfully. Results are in folder runtests_output.

```
cd ~/project_repo/tools/CTF
source activate_ctf_env.sh
../ctf_tests/run_tests.sh sca
../ctf_tests/run_tests.sh utc
../ctf_tests/run_tests.sh ft
../ctf_tests/run_tests.sh vv
```

Using an editor, examine the output in folder runtests_output.

Documents

File	Modified
PDF File CTF_SUG.pdf	Jan 18, 2023 by Larry Harris
PNG File image-2023-10-17_15-16-58.png	Oct 17, 2023 by Larry Harris
PNG File image-2023-10-17_15-20-23.png	Oct 17, 2023 by Larry Harris
PNG File image-2023-10-17_15-21-11.png	Oct 17, 2023 by Larry Harris
PNG File image-2023-10-17_15-23-2.png	Oct 17, 2023 by Larry Harris
PNG File image-2023-10-17_15-24-55.png	Oct 17, 2023 by Larry Harris

Drag and drop to upload or [browse for files](#) 
[Download All](#)