Open STEM ware

“Hello World”
NASA Style Coding Lessons

Basecamp Version 2.6

July 2025

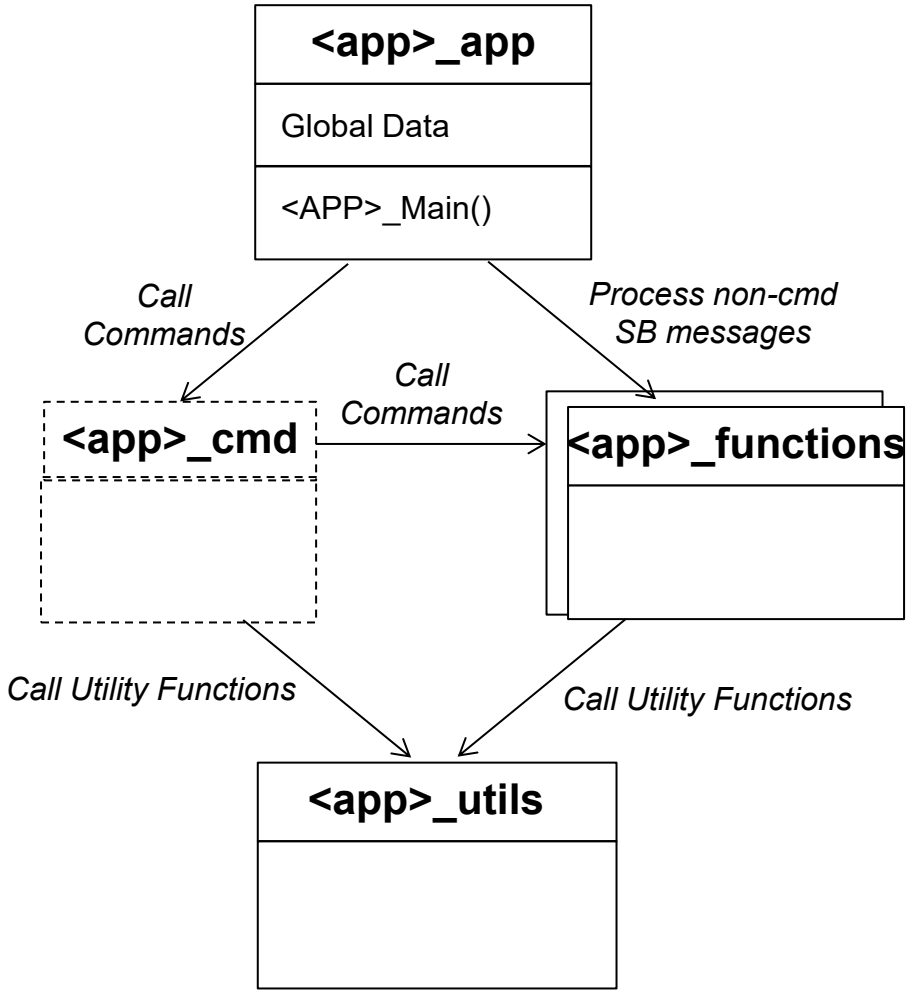
Introduction

- **These slides provide guidance for doing the NASA Sample Hello coding tutorial exercises**
- **The "Sample" tutorials teach the basics of creating an app using the NASA app design style**
- **The Sample Hello app template creates a minimal cFS application**
  - The application design follows the NASA app design conventions described in the next few slides
  - The initial hello app is a pared down version of NASA's Sample App https://github.com/nasa/sample_app
  - The coding exercises introduce developers to the different app components that result in an app that includes all of the features of the Sample App.
- **Prerequisites**
  - Working knowledge of the C programming language
  - Familiarity with Basecamp's GUI operations covered by the built-in introduction tutorial
  - Basic understanding of flight software context, the cFS architecture, and the cFS Application Developer's Guide and Basecamp's Application Developer's Guide

NASA App Design Style (1 of 2)

- **The NASA app designs don't follow a rigid design pattern but they do have similar design structures**
- **The main app file defines a global data structure that is accessed by functions that can reside in any of the app's source files**
  - These apps were designed when onboard memory and processor speeds were significantly constrained
  - Sharing global memory reduces memory footprints and avoids excessive memory copying
  - Global memory can also simplify in-orbit patches
- **The main app file contains**
  - The app's entry point called by Executive Services and the app's initialization function that registers with cFE services
  - The app's main loop of execution
  - Housekeeping telemetry generation
  - The no operation and reset counter commands
  - Apps are less consistent once beyond these basic functions
- **Functions typically contained in files outside of main are invoked by commands and software bus messages**
- **Note the cFS Framework does not dictate a particular app design strategy**
  - Basecamp apps use an object-based design implemented in C
  - There are examples of C++ apps within the cFS community

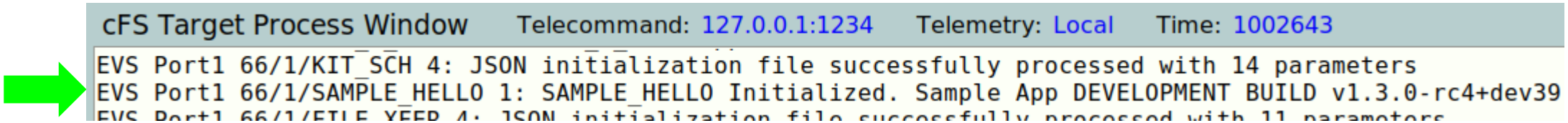
NASA App Design Style (2 of 2)



- Global memory can be read and written by any module (not shown)
- Not all apps have an <app>_cmd file and command dispatching is performed in the main file

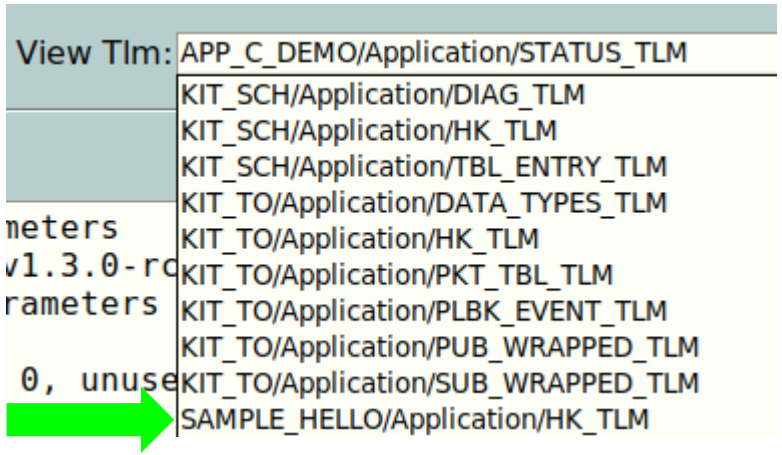
Sample Hello Operations (1 of 3)

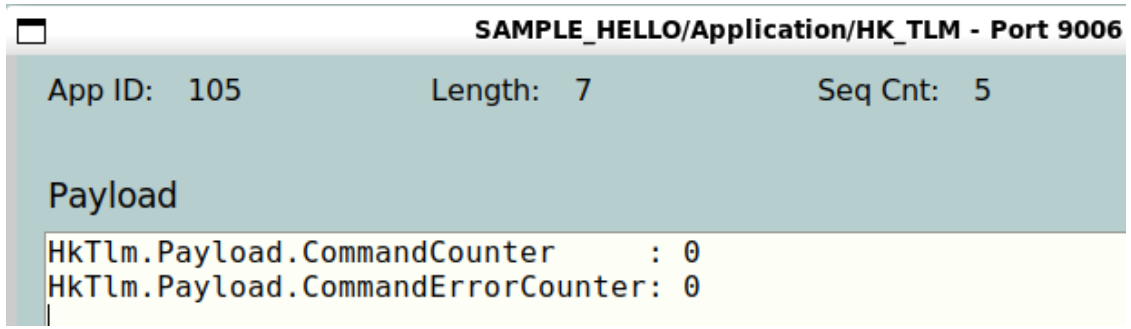
- **After generating Sample Hello, start the cFS target using the cFS <Start> button**
  - Scroll up in the cFS Target Process Window and you should see the following event message indicating the Sample Hello app successfully started



Sample Hello Operations (2 of 3)

- **Open the Sample Hello status telemetry message**
  - It only contains the valid and invalid command counters

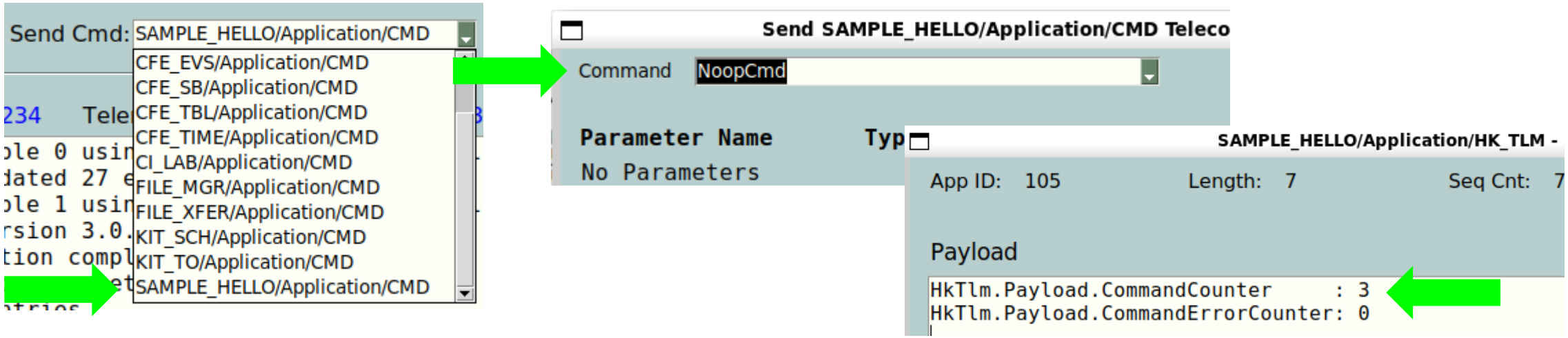




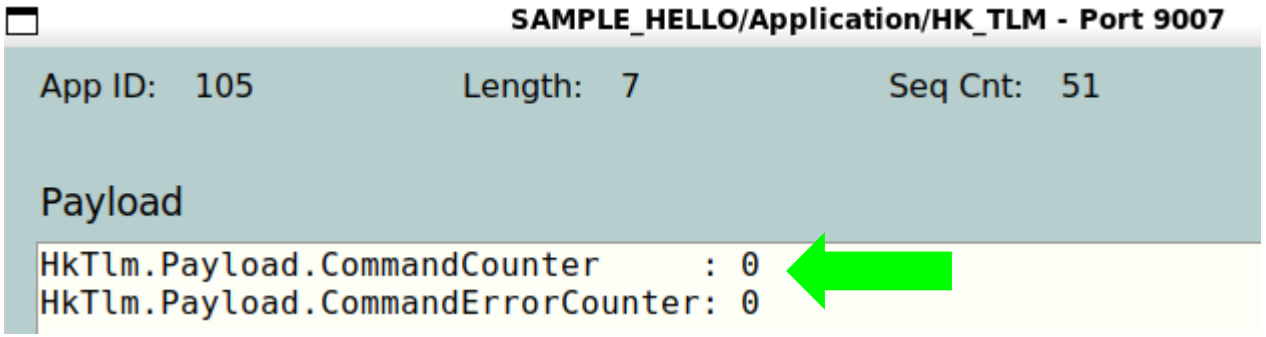
- **Status telemetry is sent every 4 seconds**
  - The app subscribes to receive the Scheduler App's 4 sec message BC_SCH_4_SEC_TOPICID

Sample Hello Operations (3 of 3)

- **Issue multiple Sample Hello Noop commands**



- **Issue a Sample Hello Reset App command to clear the command counters**



Documentation Convention

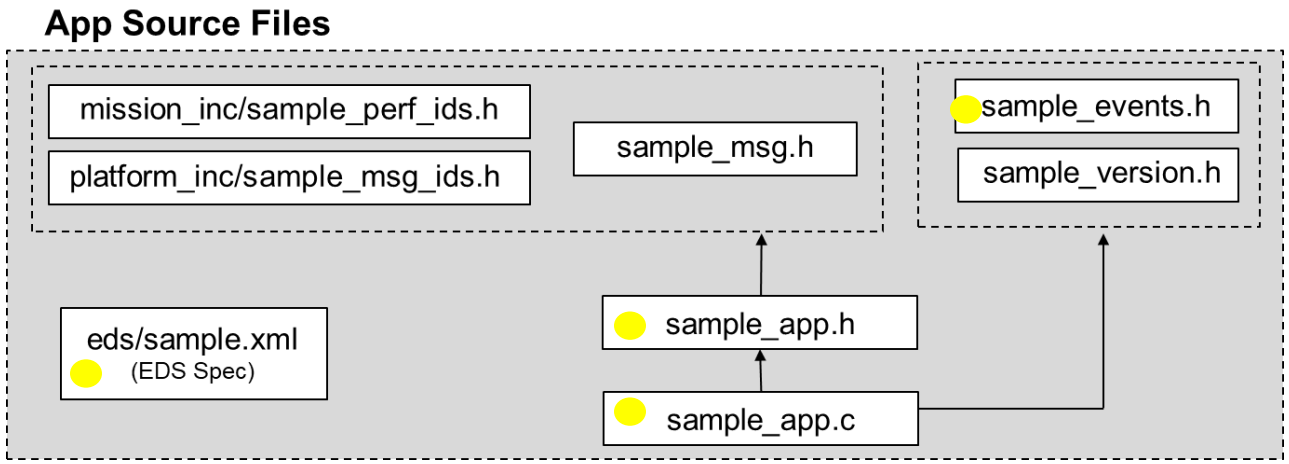
For the remaining slides the app name prefix of SAMPLE is used. This keeps the name short and consistent. When apps are generated from templates the user can use any name they choose.

Lesson 1 – Add a Command (1 of 3)

Objectives

- Learn how to define commands using Electronic Data Sheets
- Learn how to dispatch a command using its function code

The following files are modified in this lesson



The new command is added to the main app C file to minimize the scope of changed files. In practice, the command function could be in a separate file that is related to the command's function.

Lesson 1 – Add a Command (2 of 3)

sample.xml

- The new command is defined in two parts
- The *SetParam_CmdPayload* defines the command parameter
  - The Developer's Guide explains the naming convention
  - BASETYPES is an EDS package defined in the cFE EDS specs

sample_msg.h

- The new command requires a new command function code and by convention the macro names en din _CC

sample_events.h

- The new command requires a new event message identifier and by convention the macro names end in _EID

sample_app.h

- The new command function protype is added

sample_app.c

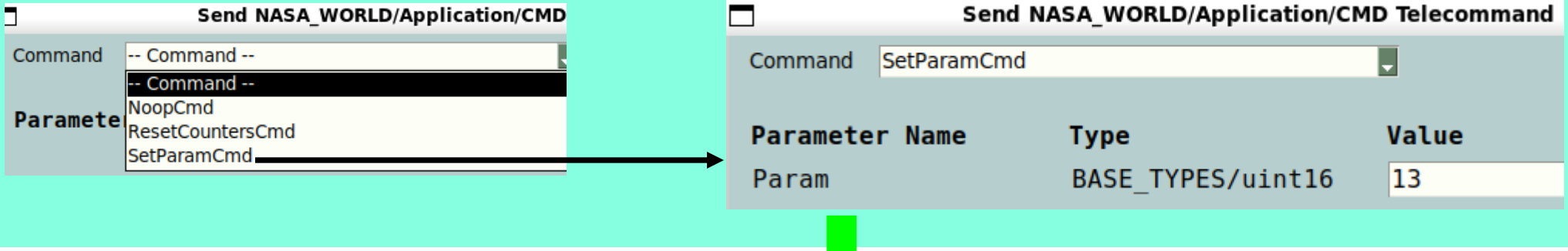
- A call to the new command function is added to the SAMPLE_ProcessGroundCommand()'s

Lesson 1 – Add a Command (3 of 3)

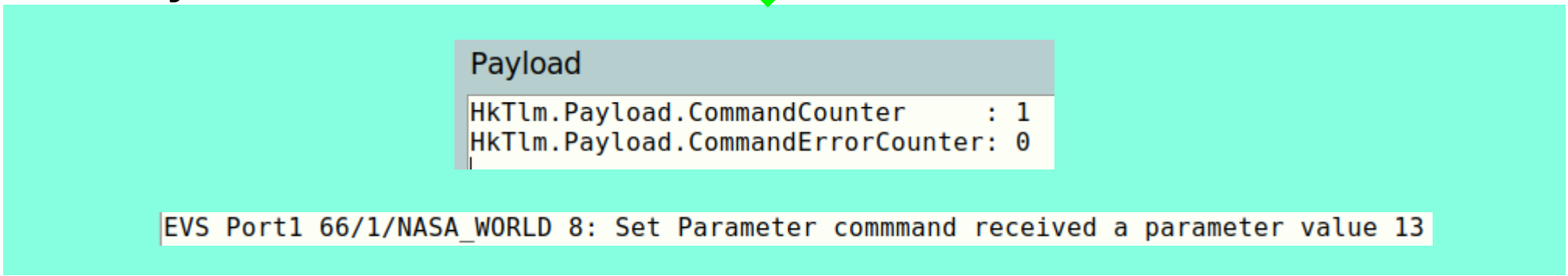
Verification

1. Use the main screen's cFS Build button to build the target (only existing files changed)
2. Since the EDS was modified, the GUI ,must be restarted to use the new command
3. Verify the new code by sending the new command, observing the event message and valid command counter

Telecommand



Telemetry

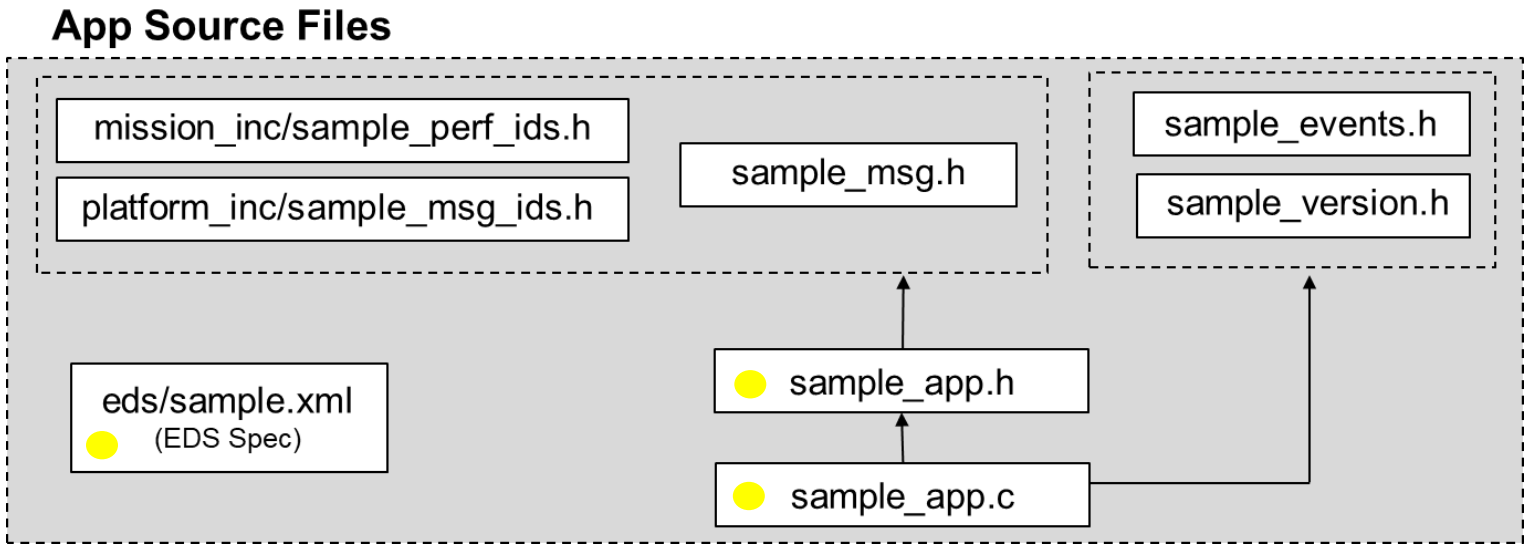


Lesson 2 – Add Telemetry Data Point (1 of 3)

Objectives

- Learn how to define telemetry messages using Electronic Data Sheets
- Introduce the concept of an app ”Housekeeping Cycle”

- **The following files are modified in this lesson**



Lesson 2 – Add Telemetry Data Point (2 of 3)

sample.xml

- Since the *HkTlm_Payload* container type already exists, this change only requires a new *<EntryList>* entry

sample_app.h

- A new variable needs to be added to save the command parameter so it can be sent in telemetry

sample_app.c

- The new command parameter variable needs to be
  - Initialized in the app initialization function
  - Set in the set command parameter function
  - Copied to the housekeeping packet

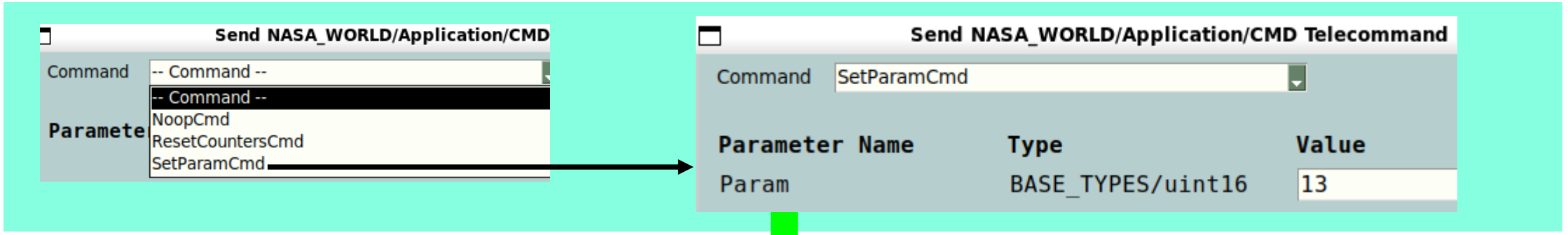
Notes

- The *SAMPLE_HkTlm_t* structure is generated by the EDS toolchain
- The app's execution period when it sends its housekeeping telemetry is often referred to as the "housekeeping cycle"
- Apps often perform other low frequency activities in this housekeeping cycle such as table validation as you'll see in the NASA Table tutorial

Verification

1. Use the main screen's cFS Build button to build the target (only existing files changed)
2. Since the EDS was modified, the GUI ,must be restarted to use the new telemetry
3. Verify the new code by sending the set parameter command and observing the telemetry is updated with the commanded value

Telecommand



Telemetry