Open STEM ware

cFS Basecamp
Hello World Coding Lessons

Version 2.6

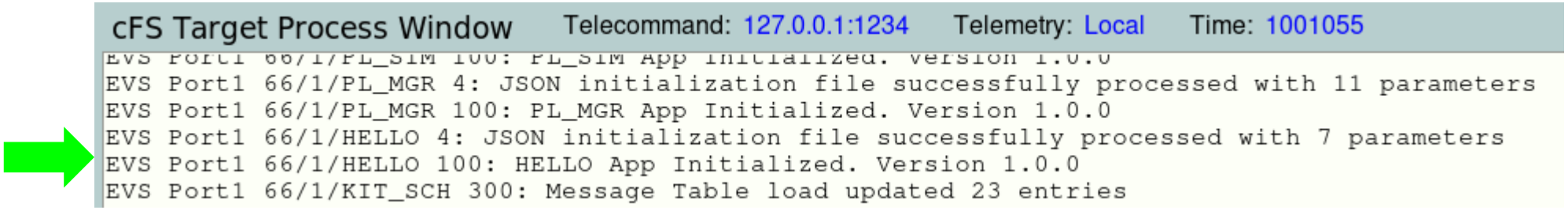
July 2025

Introduction

- **These slides provide guidance for doing the Hello World coding tutorial exercises**
- **The "Hello App Designs" section in Basecamp's *Application Developer's Guide* provides design information for all of Hello App coding tutorials**
  - Having all of the design information in one place makes the developer's guide flow better
  - It should be used in conjunction with this guide
- **The Hello World app template creates a minimal cFS application**
  - The application design follows the Basecamp app design conventions as described in the developer's guide
  - The coding exercises introduce developers to the different app components
- **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

Hello World Operations (1 of 3)

- **After generating Hello World, start the cFS target using the cFS <Start> button**
  - Scroll up in the cFS Target Process Window and you should see the following two event messages that indicate the Hello World app successfully started

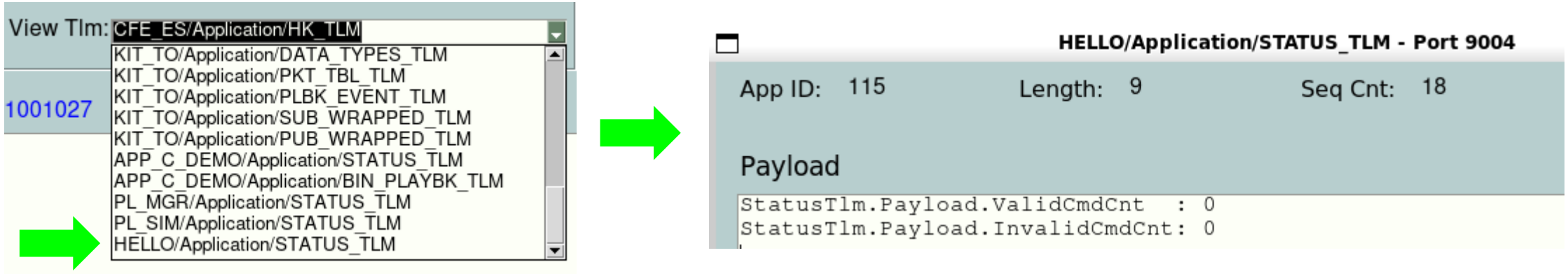


- **Use the File Browser to download hello_ini.json and open it in the text editor**
  - Note the TOPICID values are populated when “make topicids” is executed

```
{
   "title": "Hello initialization file",
   "description": ["Define runtime configurations"],
   "config": {

      "APP_CFE_NAME": "HELLO",
      "APP_PERF_ID":  127,

      "APP_CMD_PIPE_DEPTH": 5,
      "APP_CMD_PIPE_NAME":  "HELLO_CMD",

      "HELLO_CMD_TOPICID": 6247,
      "BC_SCH_4_SEC_TOPICID": 6228,
      "HELLO_HK_TLM_TOPICID": 2163

   }
}
```

Hello World Operations (2 of 3)

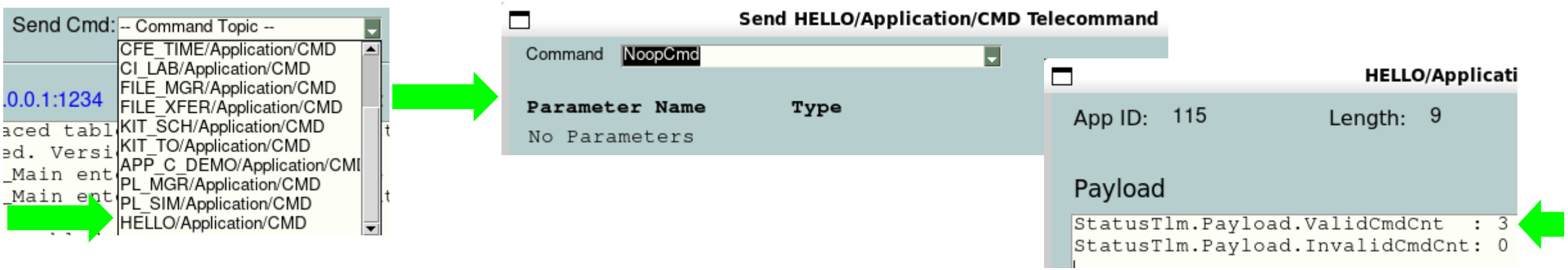
- **Open the Hello World status telemetry message**
  - It only contains the valid and invalid command counters



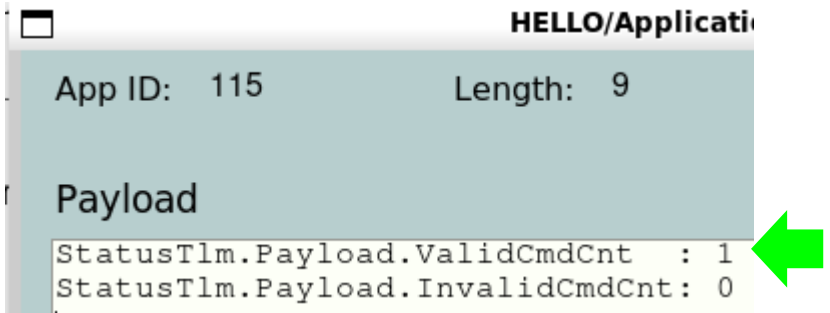
- **Status telemetry is sent every 4 seconds**
  - The app subscribes to receive the Scheduler App's 4 sec message BC_SCH_4_SEC_TOPICID
  - BC_SCH_4_SEC_TOPICID is a parameter in the JSON init file so it can be referenced by the app

Hello World Operations (3 of 3)

- **Issue multiple Hello World Noop commands**
  - Note the *Quick Cmd* menu includes a *"Noop/Reset App"* entry that may be more convenient than *Send Cmd*



- **Issue a Hello World Reset App command to clear the command counters**
  - Note Basecamp's convention is to have the reset command clear the command counters and then the counter increments to 1 because the reset command was successful

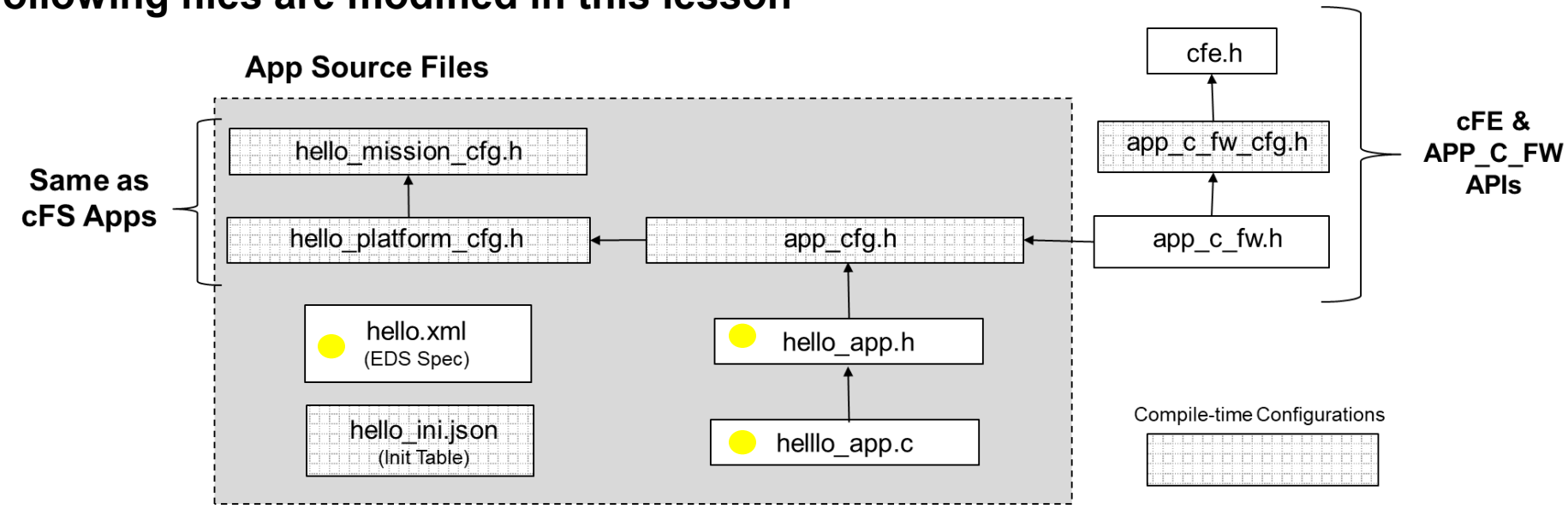


Lesson 1 – Add a Command (1 of 3)

Objectives

- Learn how to define commands using Electronic Data Sheets
- Learn how to use APP_C_FW's Command Manager utility for registering and dispatching commands

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



Typically, new commands are not added to the main app file because in Basecamp's object-based design strategy new commands would be part of an object owned by the main app. It is being done here to teach the objectives while keeping the app structure simple.

Lesson 1 – Add a Command (2 of 3)

hello.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
- *APP_C_FW/APP_BASE_CC* is the starting Command Code for apps
  - The Noop and Reset commands are in every app so the framework defines their command codes

hello_app.h

- The new command requires a new event message identifier and by convention the macro names end in _EID
- XXX_SetParamCmd() definition must follow the APP_C_FW Command Manager API requirements

hello_app.c

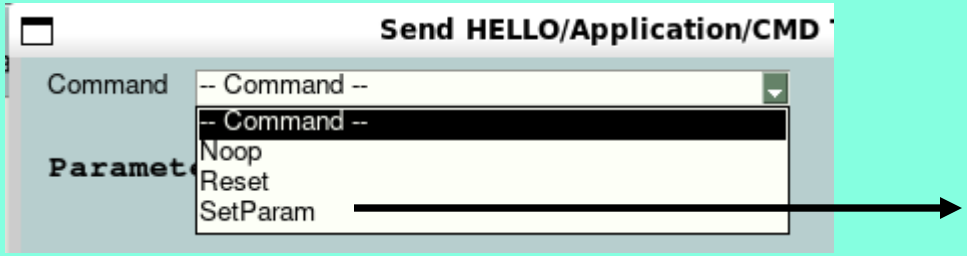
- The const *CmdPayload definition in XXX*CmdPayload() is a coding idiom that makes accessing command parameters easy and consistent
- CMDMGR_RegisterFunc() registers the command processing function which is invoked because of the call to CMDMGR_DispatchFunc() in ProcessCommands()

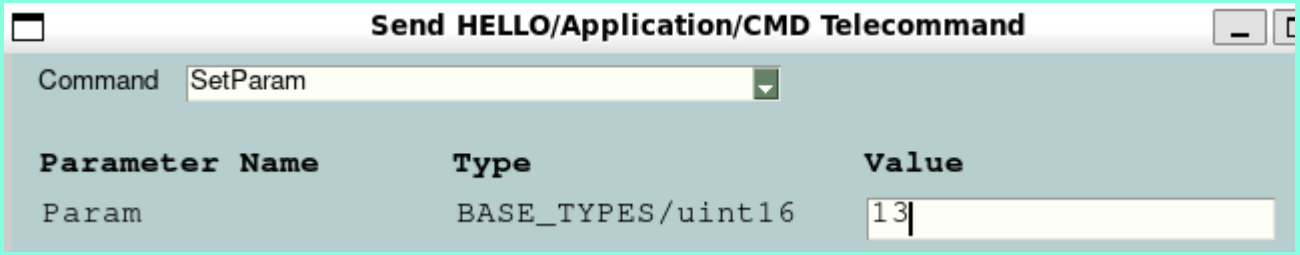
Lesson 1 – Add a Command (3 of 3)

Verification

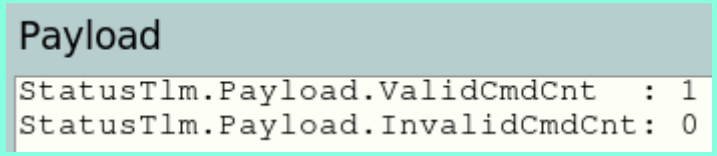
1. Use the main screen's cFS Build button to build the target (only existing files changed)
2. Since the EDS was modified, the GUI ,must be restarted to use the new command
3. Verify the new code by sending the new command, observing the event message and valid command counter

Telecommand





Telemetry



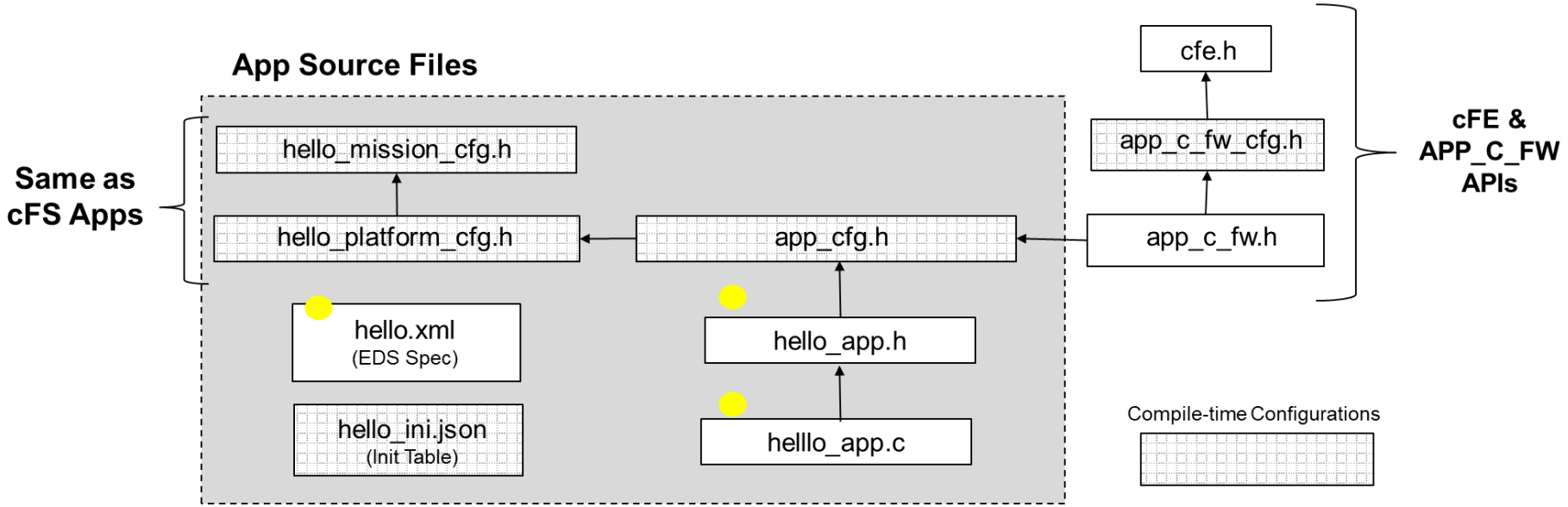

```
EVS Port1 66/1/HELLO 104: Set Parameter commmand received a parameter value 13
```

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

Objectives

- Learn how to define telemetry messages using Electronic Data Sheets
- Introduce how telemetry messages are managed in a Basecamp app

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



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

hello.xml

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

hello_app.h

- A new variable needs to be added to save the command parameter so it can be sent in telemetry
- This variable is added to the hello world app's object data
- As mentioned in lesson one, this new command would typically be part of an object owned by the main app and it is being done here to keep the exercise simple

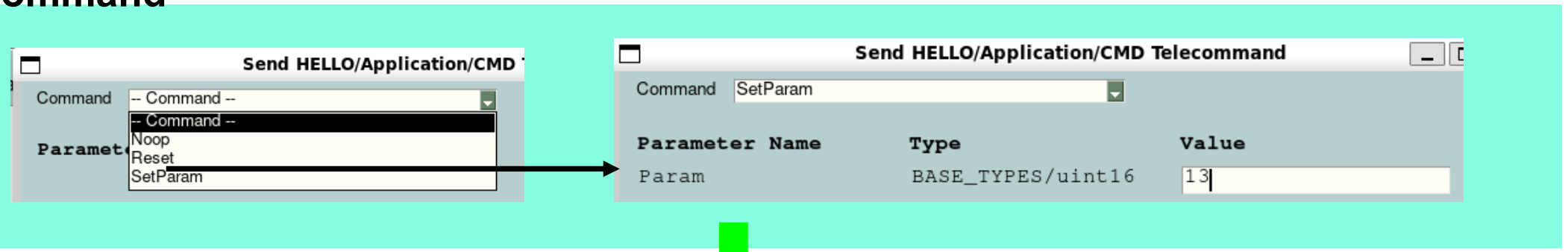
hello_app.c

- The XXX_SetParamCmd() function needs to be modified to save the parameter value
- The SendStatusTlm() needs to be modified to copy the command parameter into the telemetry packet

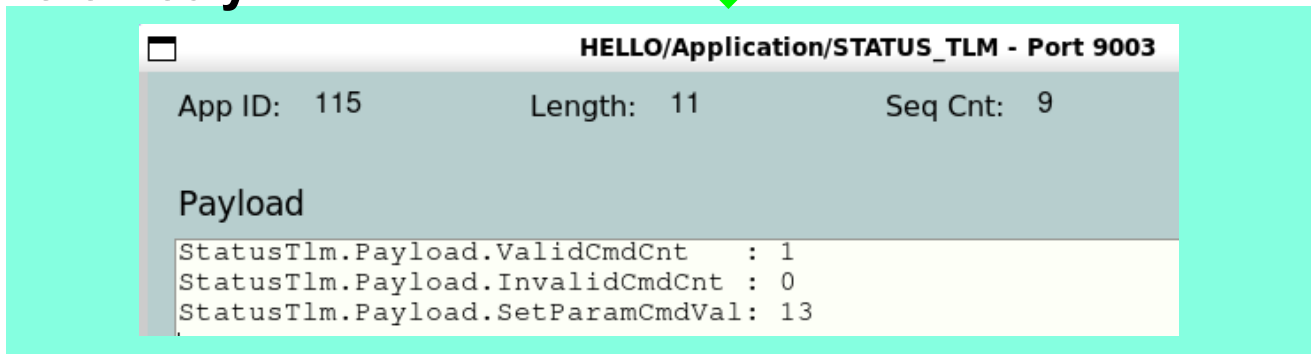
Verification

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

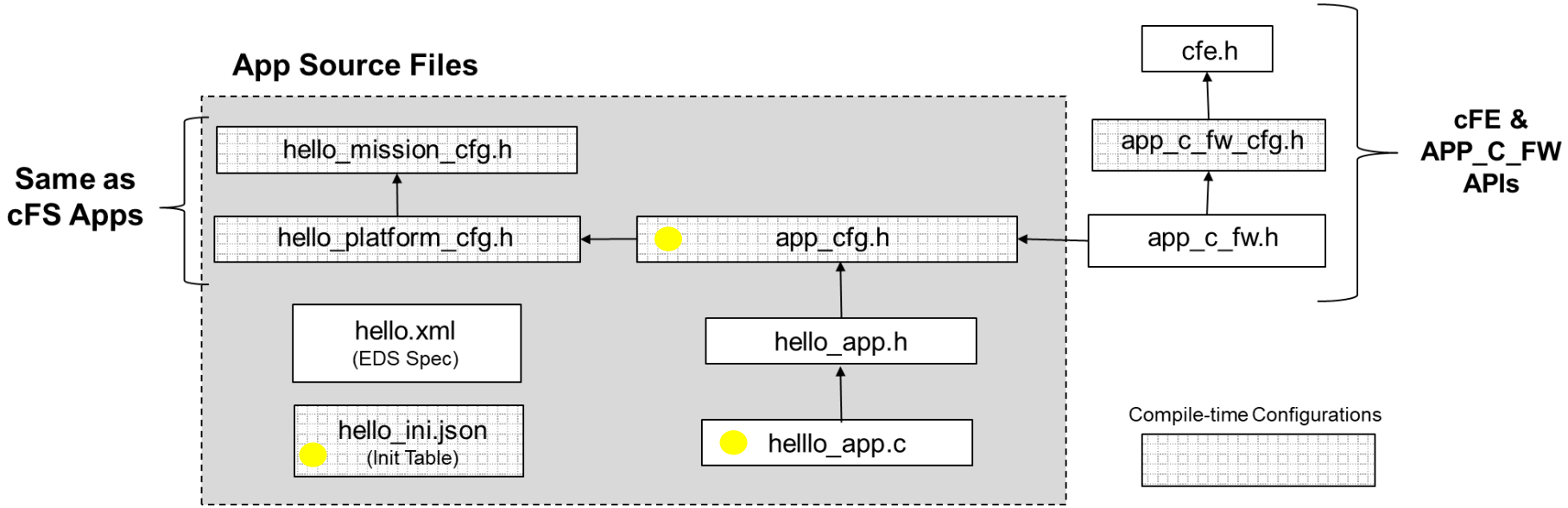


Lesson 3 – Rename JSON Init Parameter (1 of 3)

Objectives

- Introduce Basecamp's app parameter initialization JSON files
- Learn how init parameters are defined and accessed

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



Lesson 3 – Rename JSON Init Parameter (2 of 3)

cpu1_hello_wrld.ini

- Changing the APP_CMD_PIPE_DEPTH parameter name does not effect the how the parameter is used
- The value of 7 is chosen because it is not used by any other apps and will be used as part of the verification
- During the target build process this file is copied from basecamp_defs to the build/exe/cpu1/cf and the ‘cpu1_’ prefix is removed from the filename

app_cfg.h

- This JSON initialization file parameter names are defined in this file, block comment has detailed instructions
- Initialization parameters can either by of type string or integer

hello_app.c

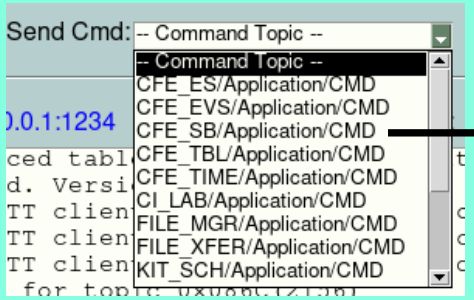
- The initialization parameters values are retrieved using either the INITBL_GetIntConfig() or INITBL_GetStrConfig() functions
- APP_CMD_PIPE_MAX is used in the CFE_SB_CreatePipe() call and defines the maximum number of packets that can be on the pipe

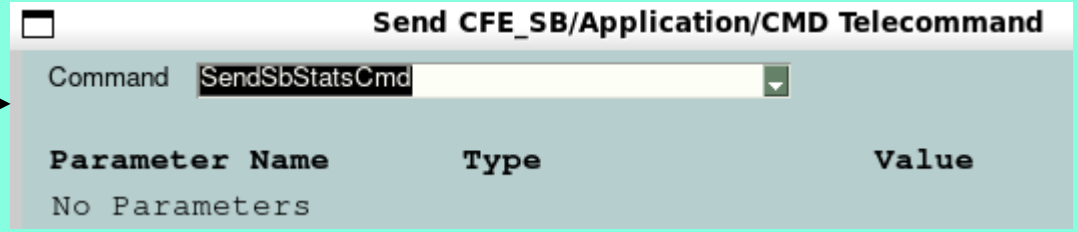
Lesson 3 – Rename JSON Init Parameter (3 of 3)

Verification

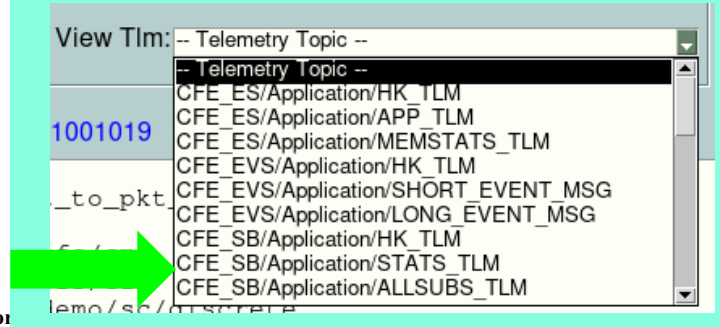
1. The Software Bus Stats telemetry message will be used to indirectly verify the initialization parameter
2. Open SB's STATS_TLM using the "View Tlm" drop down menu
3. Issue a *SendSbStatsCmd* which causes SB to send one STATS_TLM message. Scroll down until you find an app with a MaxQueueDepth of 7 which implies this is the Hello World app. You can change the parameter value again and follow these steps to verify you find an app with the expected value.

Telecommand





Telemetry




```
StatsTlm.Payload.PipeDepthStats[6].PipeId.BaseType    : 34996231
StatsTlm.Payload.PipeDepthStats[6].MaxQueueDepth      : 7
StatsTlm.Payload.PipeDepthStats[6].CurrentQueueDepth  : 0
StatsTlm.Payload.PipeDepthStats[6].PeakQueueDepth     : 1
```