Lesson 2

Objectives

- Describe the Hello World App's functionality
- Describe the coding tutorial framework used by the Hello app templates
  - The coding tutorial documentation is designed to standalone. Some of the information in the Hello World Coding document is repeated here so the tutorial design can be explained.

Notes

1. This lesson serves as an introduction to app designs, for more information refer to the Basecamp Application Developer's Guide and to the other "Hello-*" app template coding tutorials

Hello World App Functionality

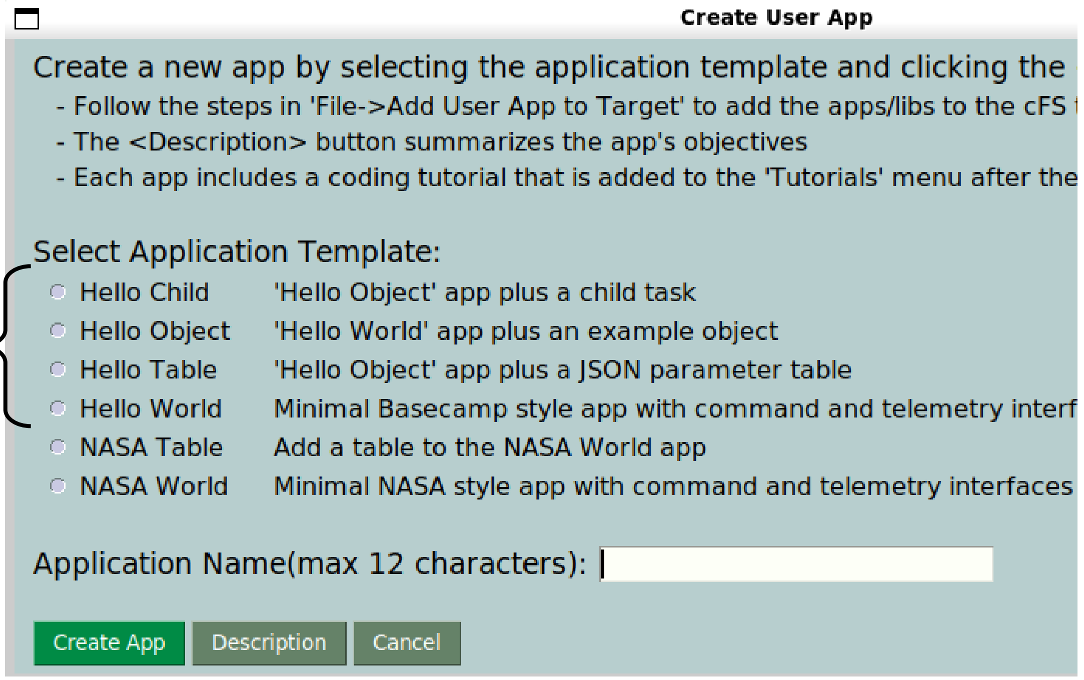
- **The Hello World app implements the minimal functionality required by an app**
  - Create a Software Bus “Pipe” and register to receive messages
  - Accept command messages and execute command-specific functions
  - Output status telemetry
- **Some functions are “NASA/Goddard design patterns” that have evolved based on experience with Low Earth Orbit (LEO) satellites**
  - If the app successfully initializes, send an event message identifying the app version
    - Provide evidence that each app has successfully started and it’s the expected version
  - Provide command valid and command invalid counters in periodic status telemetry
    - Allows the ground operators to confirm that a command was received and processed with either a successful or unsuccessful outcome
  - Send a “housekeeping” telemetry message at a constant periodic rate
    - Housekeeping is a NASA/Goddard colloquial term that means status
    - Allows command counters to be checked after sending a command
  - Provide a “No Operation (NOOP)” command that increments the command valid counter and sends an event message containing the app version
    - Allows the ground operators to confirm the communication path to an app is operational and that the app is functioning properly
  - Provide a “Reset Counters” command that clears the command counters

Basecamp's Minimal App Functionality

- **Basecamp apps include the NASA/Goddard design patterns with a few additions**
- **Basecamp apps use Basecamp's Application C Framework (APP_C_FW)**
  - Provides application services and utilities that support object-based designs
  - Developers can focus on developing app functional objects
- **Define command and telemetry messages using Electronic Data Sheets**
- **Use a JSON initialization parameter file to define runtime configurations**
  - cFS target management tools can modify these files that allows automated system integration
  - Read during an app's initialization
  - Many mission and platform configurations traditionally defined in C header files can be defined in this initialization file
- **APP_C_FW Command Manager**
  - Apps register each object's command functions with the Command Manager
  - When a command message is received, Command Manager calls the corresponding command function
- **The Reset Counter command is called a Reset App and has a broader scope than just resetting counters**
  - The Reset App command results in an app's status being reset to an app-specific default state
  - Each object within an app provides a reset function that is called
  - If a status item is effected by the reset command then it should be included in a periodic telemetry message so the new status can be verified

Hello App Coding Tutorials (1 of 6)

- **Basecamp's "Hello *" series of create app templates include coding tutorials**



Create User App

Create a new app by selecting the application template and clicking the

- Follow the steps in 'File->Add User App to Target' to add the apps/libs to the cFS
- The <Description> button summarizes the app's objectives
- Each app includes a coding tutorial that is added to the 'Tutorials' menu after the

Select Application Template:

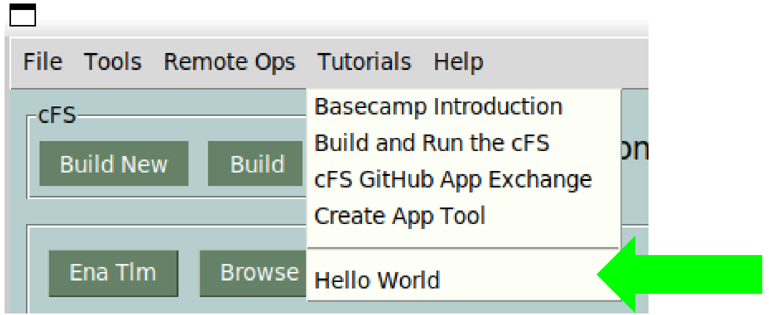
Hello Child	'Hello Object' app plus a child task
Hello Object	'Hello World' app plus an example object
Hello Table	'Hello Object' app plus a JSON parameter table
Hello World	Minimal Basecamp style app with command and telemetry interf
NASA Table	Add a table to the NASA World app
NASA World	Minimal NASA style app with command and telemetry interfaces

Application Name(max 12 characters):

Hello

App Series

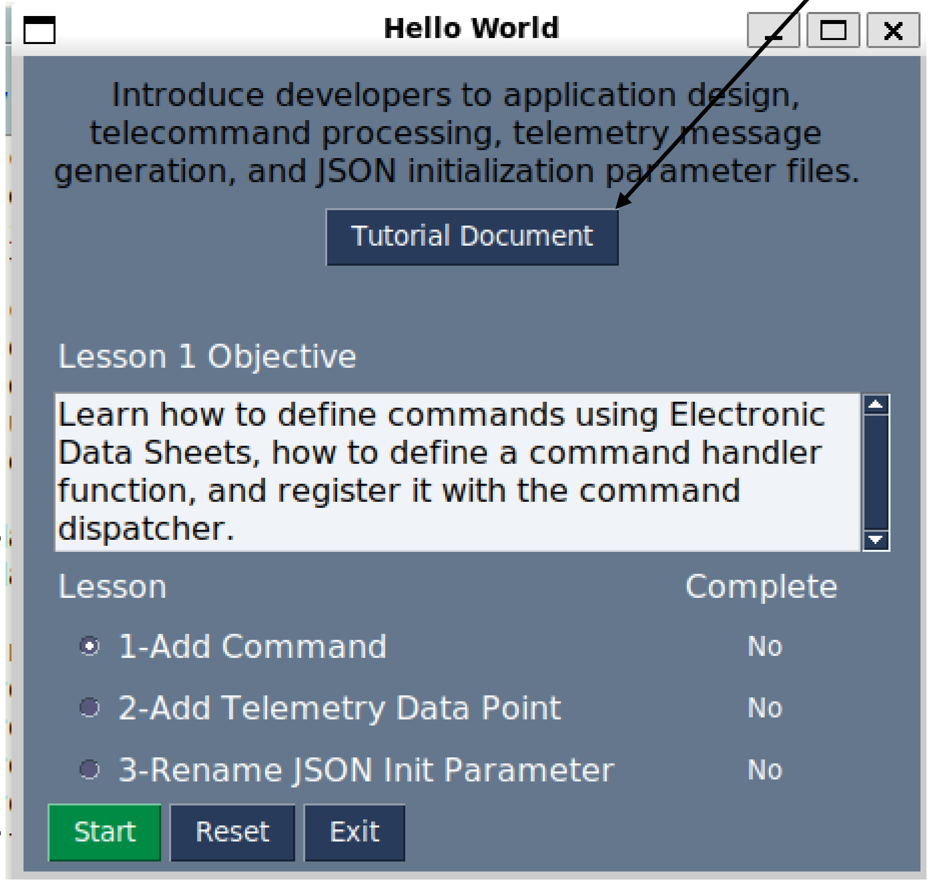
- **After an app is created and the Python GUI is restarted the coding tutorial will be listed in bottom section of the Tutorials dropdown menu**



Hello App Coding Tutorial (2 of 6)

Launch tutorial document in PDF file viewer

- Contains design information that spans all of the lessons



Hello World

Introduce developers to application design, telecommand processing, telemetry message generation, and JSON initialization parameter files.

Lesson 1 Objective

Learn how to define commands using Electronic Data Sheets, how to define a command handler function, and register it with the command dispatcher.

Lesson	Complete
[✓ 1-Add Command]	No
2-Add Telemetry Data Point	No
3-Rename JSON Init Parameter	No

Select and start individual coding lessons

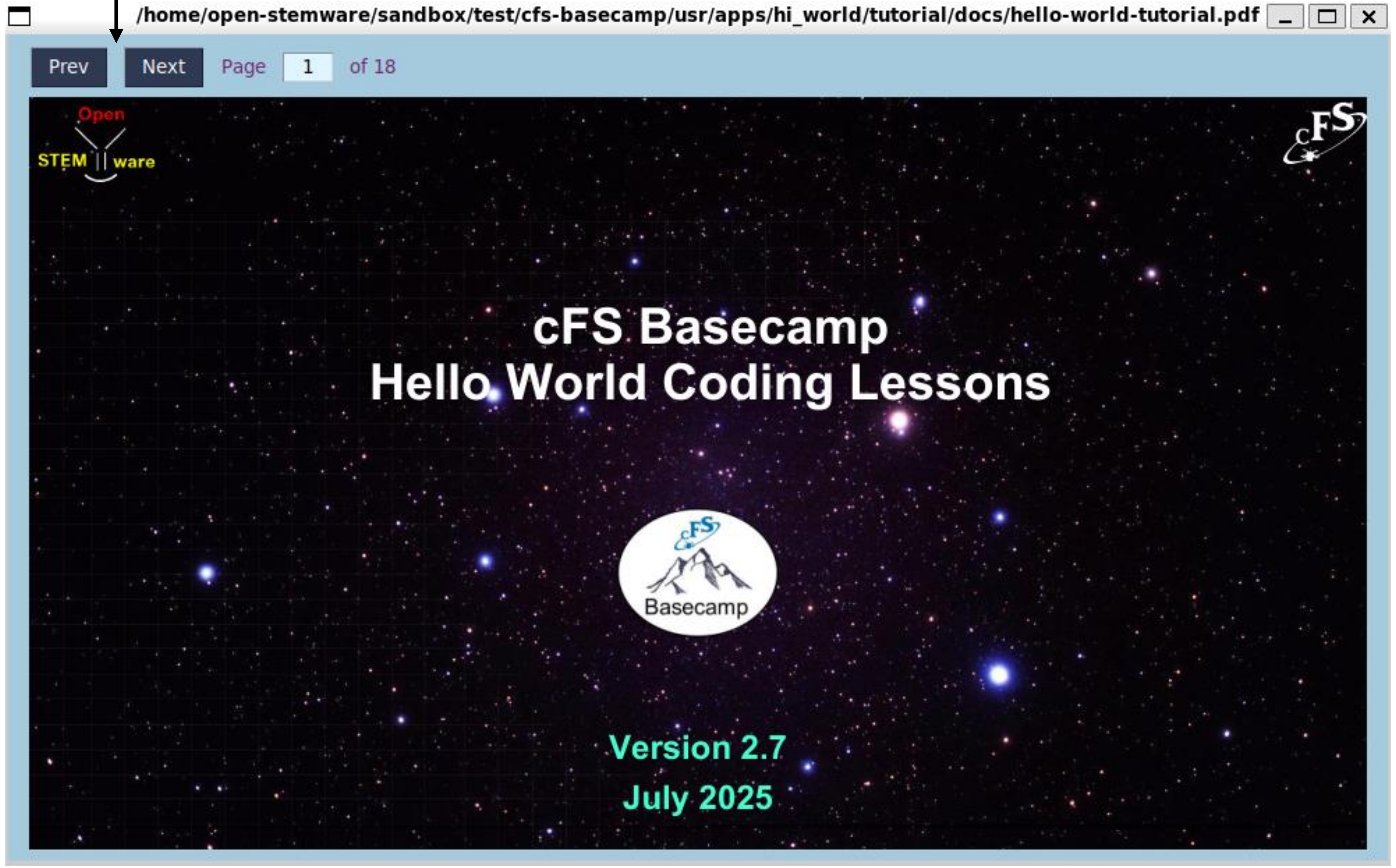
- Lessons should be done in order since each lesson builds upon previous lessons

Lessons can be marked as complete

- The completion status can be reset\
- Resetting status does not effect any code modifications

Hello App Coding Tutorial (3 of 6)

Step through the PDF tutorial document using the <Prev> and <Next> buttons

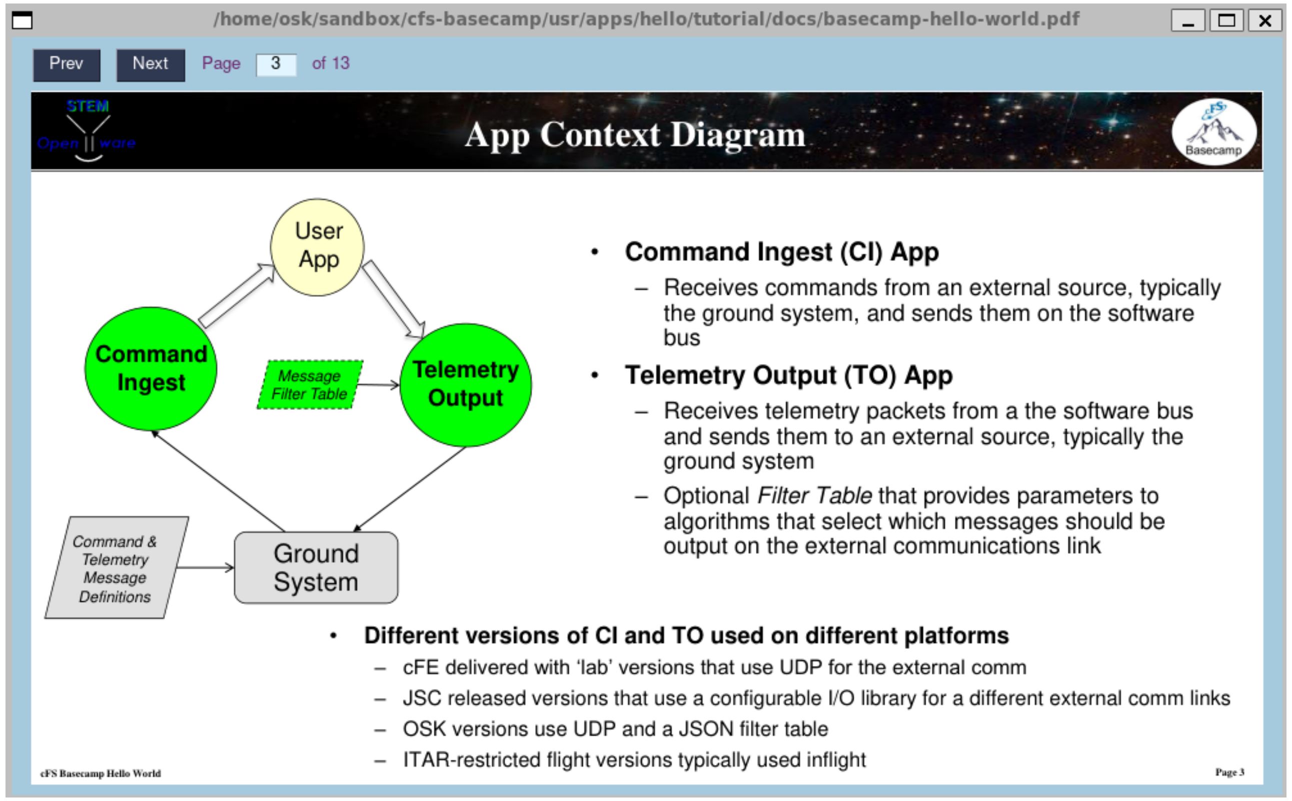

/home/open-stemware/sandbox/test/cfs-basecamp/usr/apps/hi_world/tutorial/docs/hello-world-tutorial.pdf
Prev
Next
Page 1 of 18
Open STEM ware
cFS

cFS Basecamp
Hello World Coding Lessons

Version 2.7
July 2025

Hello App Coding Tutorial (4 of 6)

- **Coding tutorials follow the same outline**
  1. Functionality and Operations
     - TBD
  2. Design
     - Provides important design information that should be understood prior to coding
  3. Coding Lessons



App Context Diagram

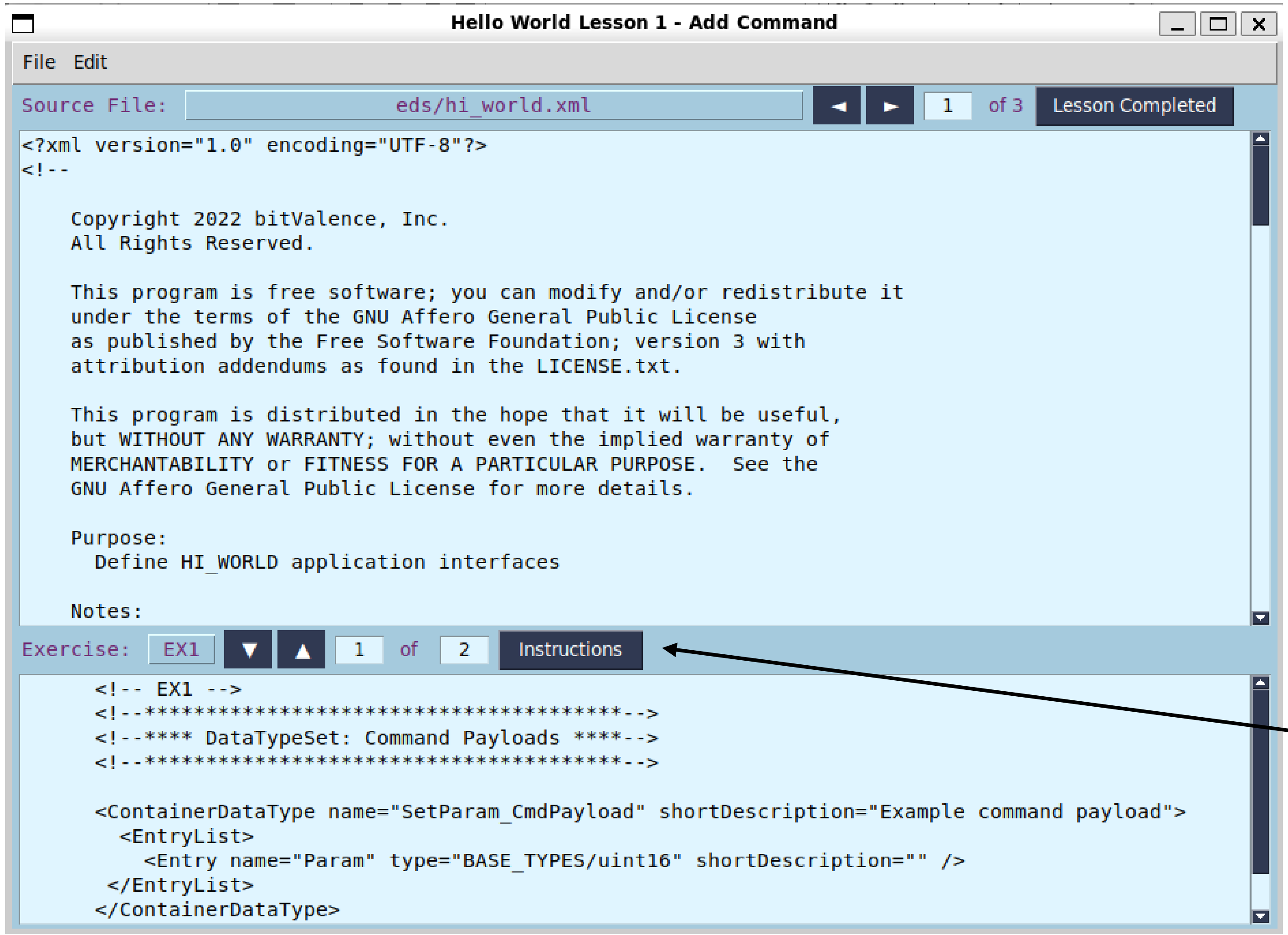
- **Command Ingest (CI) App**
  - Receives commands from an external source, typically the ground system, and sends them on the software bus
- **Telemetry Output (TO) App**
  - Receives telemetry packets from a the software bus and sends them to an external source, typically the ground system
  - Optional *Filter Table* that provides parameters to algorithms that select which messages should be output on the external communications link
- **Different versions of CI and TO used on different platforms**
  - cFE delivered with 'lab' versions that use UDP for the external comm
  - JSC released versions that use a configurable I/O library for a different external comm links
  - OSK versions use UDP and a JSON filter table
  - ITAR-restricted flight versions typically used inflight

Hello App Coding Tutorial (5 of 6)

The following GUI is launched when a lesson is started…

Source file to be edited during lesson

Solution to each exercise



```
<?xml version="1.0" encoding="UTF-8"?>
<!--

    Copyright 2022 bitValence, Inc.
    All Rights Reserved.

    This program is free software; you can modify and/or redistribute it
    under the terms of the GNU Affero General Public License
    as published by the Free Software Foundation; version 3 with
    attribution addendums as found in the LICENSE.txt.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    Purpose:
      Define HI_WORLD application interfaces

    Notes:
```

```
      <!-- EX1 -->
      <!--****************************************-->
      <!--**** DataTypeSet: Command Payloads ****-->
      <!--****************************************-->

      <ContainerDataType name="SetParam_CmdPayload" shortDescription="Example command payload">
        <EntryList>
          <Entry name="Param" type="BASE_TYPES/uint16" shortDescription="" />
       </EntryList>
      </ContainerDataType>
```

Lessons have one or more files to be edited

Lessons can be marked as complete

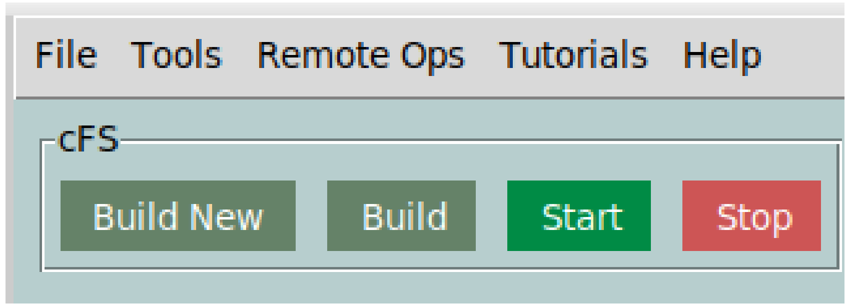
Each file can have one or more exercises

The <Instruction> button provides detailed exercise information

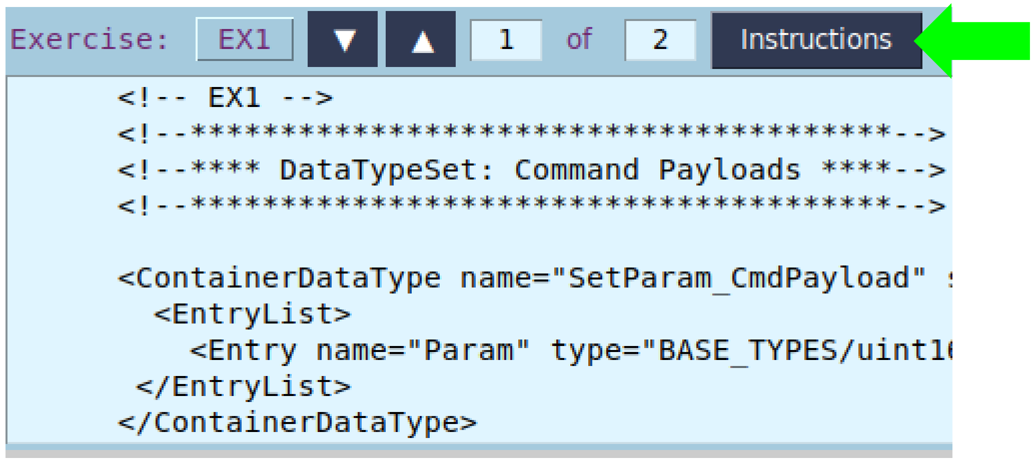
Text can be copied from solution pane and pasted into the source file pane

Hello App Coding Tutorial (6 of 6)

- **Use the main window to build and run a new cFS target**
  - *<Build New>* is used when a lesson changes an EDS definition or introduces a new file
  - *<Build>* is used when existing source files are modified. The build is typically very fast.

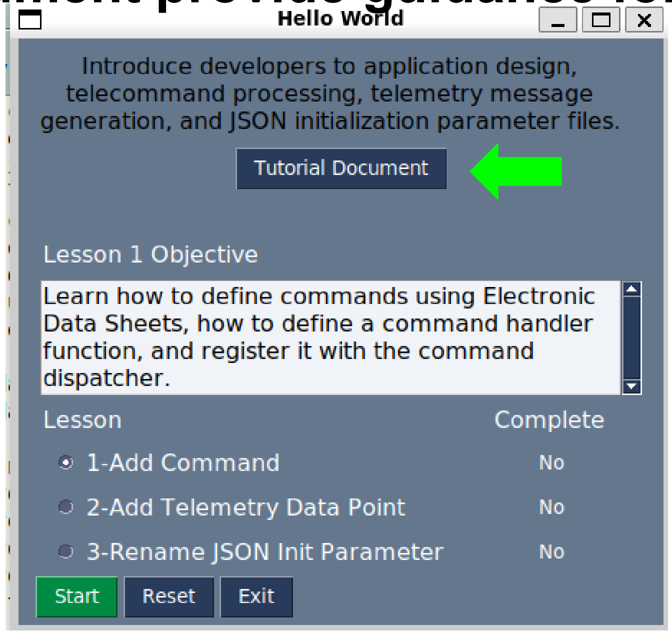


- **The Python GUI only needs to be restarted if an exercise changes an app's EDS file**
- **Lesson exercises instructions and tutorial document provide guidance for how to build and run the a new cFS target**



```
      <!-- EX1 -->
      <!--***************************************-->
      <!--**** DataTypeSet: Command Payloads ****-->
      <!--***************************************-->

      <ContainerDataType name="SetParam_CmdPayload" 
        <EntryList>
          <Entry name="Param" type="BASE_TYPES/uint1
       </EntryList>
      </ContainerDataType>
```



Introduce developers to application design, telecommand processing, telemetry message generation, and JSON initialization parameter files.

Lesson 1 Objective

Learn how to define commands using Electronic Data Sheets, how to define a command handler function, and register it with the command dispatcher.

Lesson	Complete
1-Add Command	No
2-Add Telemetry Data Point	No
3-Rename JSON Init Parameter	No