OpenSatKit Introduction Quick Start

V3.0

May 2021

Table of Contents

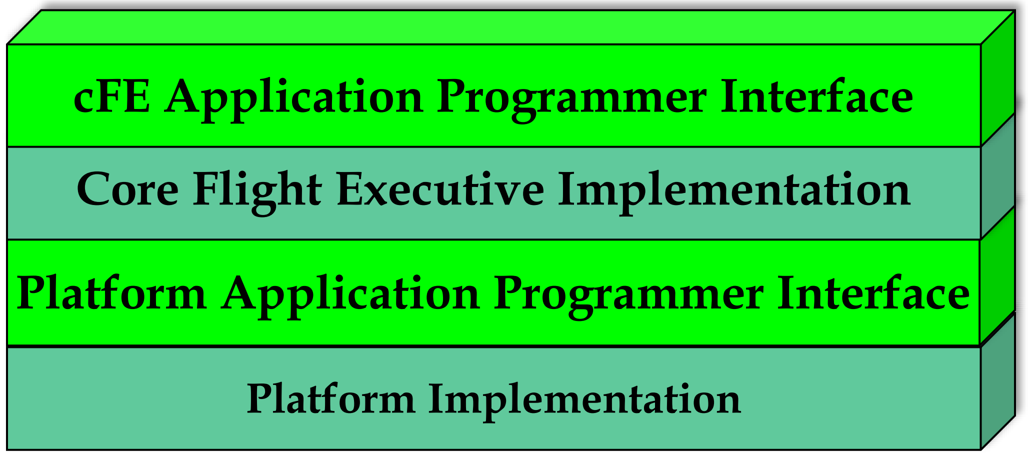
Introduction

- **The primary objectives of OpenSatKit (OSK) are to**
  1. Support cFS-based mission FSW development
  2. Provide a core Flight System (cFS) education platform environment
  3. Remotely control a cFS system on a Raspberry Pi
  4. Serve as a R&D
- **This guide provides an overview of OSK and there are separate guides for each of the four objectives**
- **The cFS is an open architecture that is designed to be ported and extended**
  - These attributes add end-user deployment/configuration complexity
  - OSK provides fully functional cFS system deployed on Linux, however…
- **OSK introduces additional complexity because it integrates two additional powerful software packages, COSMOS and the 42 Simulator, that have their own learning curve.**

The cFS provides high quality flight heritage software that implements a significant amount of mission functionality so the rewards are high if you can persist through the learning curve!

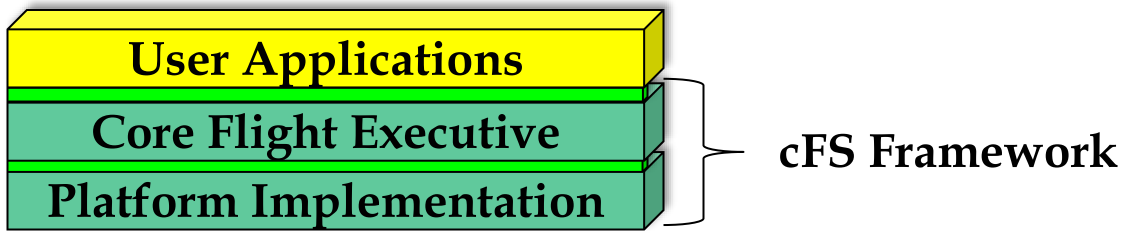
cFS Introduction

- A NASA multi-center configuration controlled open-source flight software framework


cFE Application Programmer Interface
Core Flight Executive Implementation
Platform Application Programmer Interface
Platform Implementation

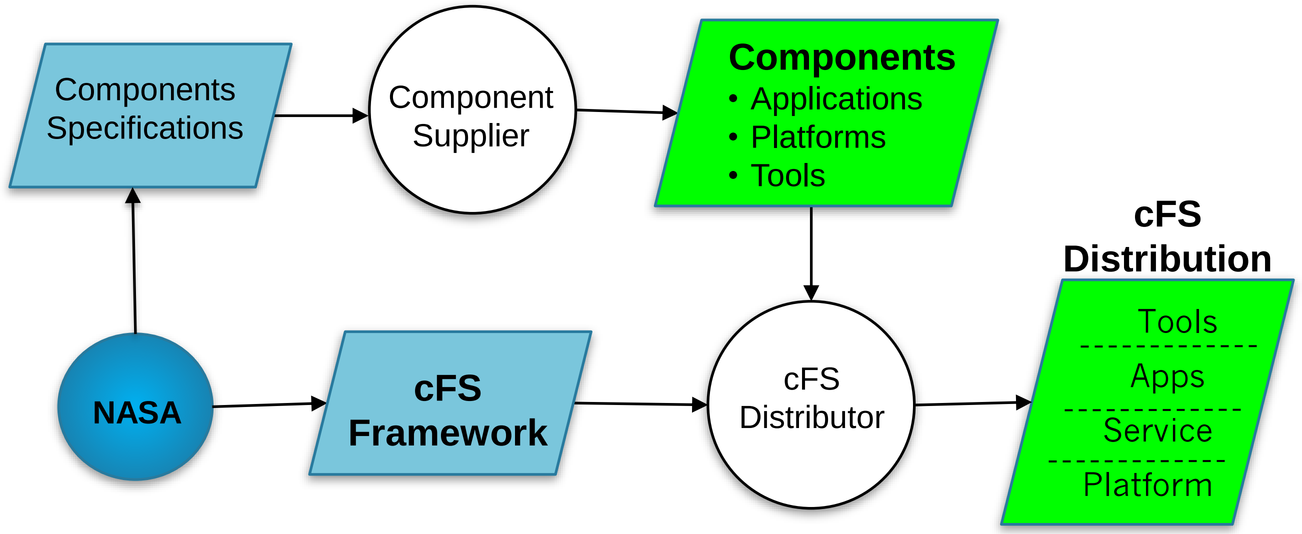
- Layered architecture with international standards-based interfaces
- Provides development tools and runtime environment for user applications
- Reusable NASA Class A/B lifecycle artifacts: requirements, design, code, tests, and documents

- The framework is ported to a platform and augmented with applications to create Core Flight System (cFS) distributions



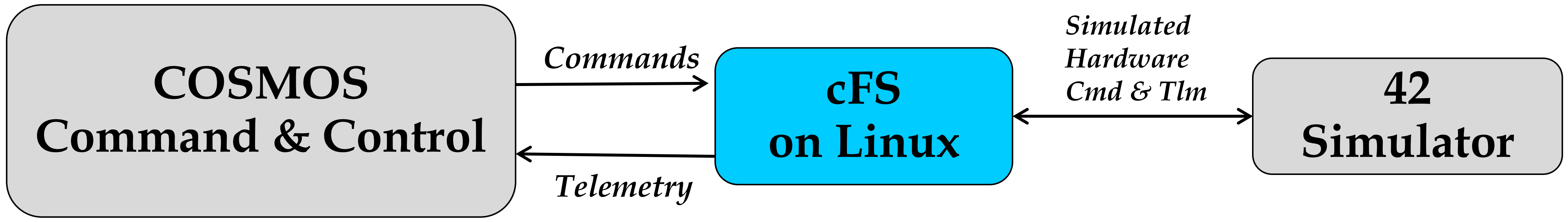
- A worldwide community from government, industry, and academia

cFS Product Model



- **A NASA multi-center configuration control board (CCB) manages releases of the open source cFS Framework and component specifications**
- **Community members (regardless of affiliation)**
  - Supply applications, platforms, and tools
  - Create cFS distributions – OSK is a distribution

OSK Architecture



- **In addition to the cFS itself, OSK uses two additional open-source applications**
  – Ball Aerospace's COSMOS command and control platform for embedded systems
  – NASA Goddard's 42 dynamic simulator
- **Each open-source package is contained in its own OpenSatKit subdirectory**

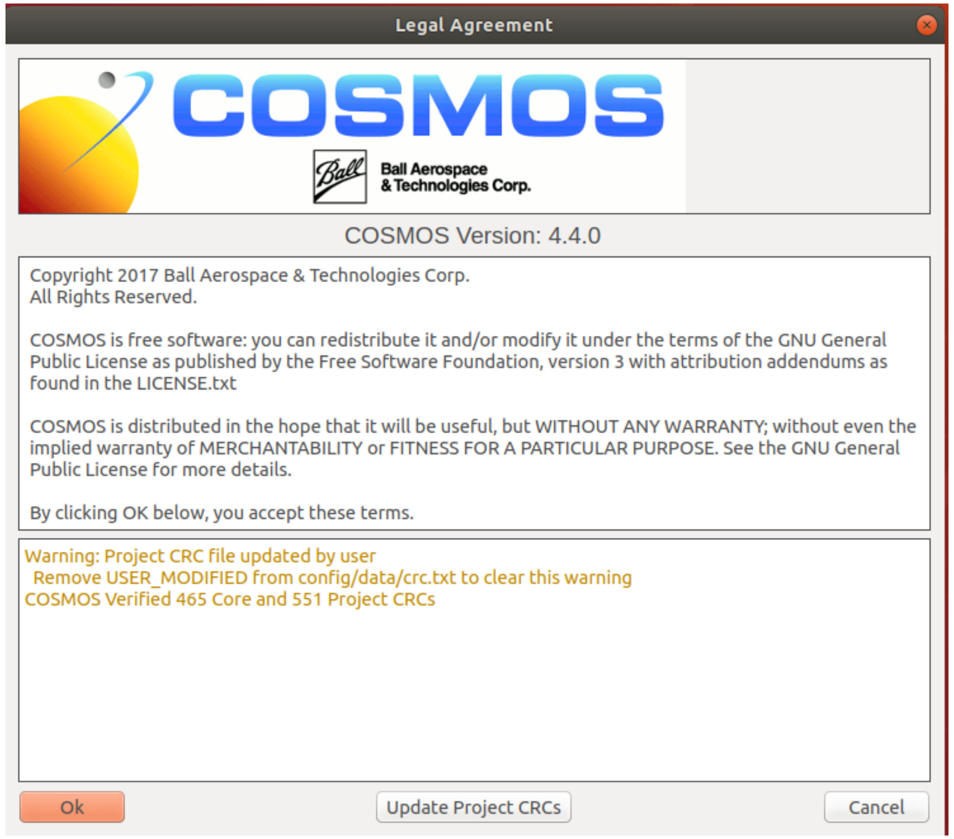
OSK Strategy

- **OSK implements extensive COSMOS configurations and customizations so OSK's screens can serve as the primary user interface for the goals listed below**
  - Doesn't preclude working with each OSK component separately
    - Develop and configure cFS from command line
    - Launch individual COSMOS tool from COSMOS launcher
    - Runn 42 simulator as a standalone application
- **OSK screens and content aligned with OSK user objectives**
- **Separate cFS targets for each user objective**
- **Learning resources use a combination of documents, screen and script-based demos, and links to YouTube videos**

2 - Running OSK

Running OSK (1 of 2)

- Open a terminal window (Ctrl-Alt-t)
- Navigate to the base directory where you installed OSK
  - "~/" is used to indicate the OSK base directory so "~/cfs" is equivalent to "/home/user/OpenSatKit-master/cfs" if OpenSatKit was installed in the home directory for an account named "user"
- Change directory to cosmos
  - cd ~/cosmos
- Start COSMOS
  - ~/cosmos$ ruby Launcher
  - You'll see a screen similar to the right.
    - Select <OK>
    - This creates the "Launcher" screen shown on the next slide



COSMOS Version: 4.4.0

Copyright 2017 Ball Aerospace & Technologies Corp.
All Rights Reserved.

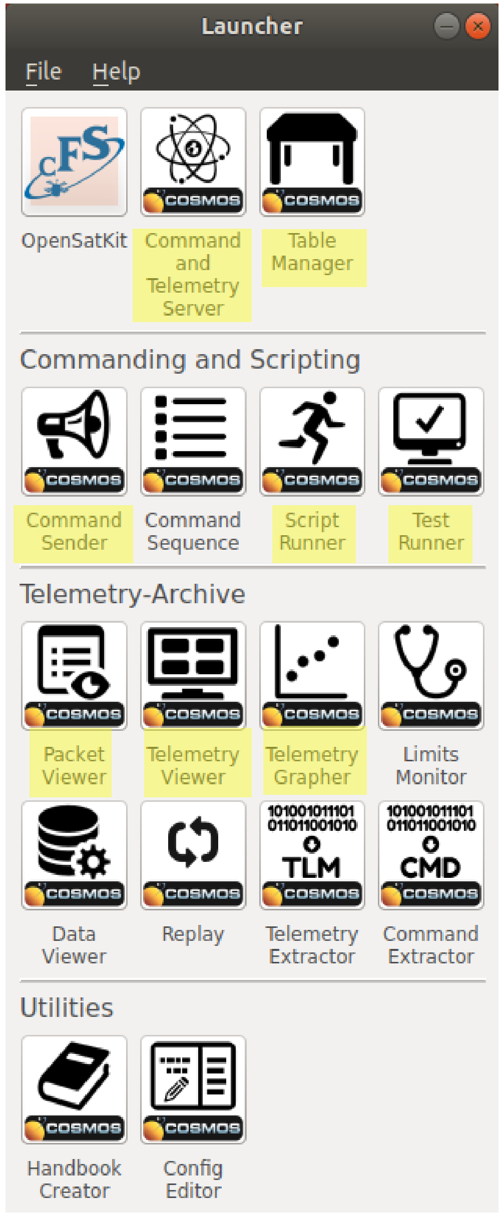
COSMOS is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, version 3 with attribution addendums as found in the LICENSE.txt

COSMOS 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 General Public License for more details.

By clicking OK below, you accept these terms.

```
Warning: Project CRC file updated by user
 Remove USER_MODIFIED from config/data/crc.txt to clear this warning
COSMOS Verified 465 Core and 551 Project CRCs
```

Running OSK (2 of 2)



- **Each tools on the COSMOS "Launcher" runs as a separate Linux process with a Graphical User Interface (GUI)**
- **Shaded tool titles indicate the primary COSMOS tools** used by OSK**
  - You do not have to invoke these tools directly
  - OSK screens launch COSMOS tools as they are needed to perform a task
  - A backup slide shows a COSMOS architectural view with the data flows between tools
- **Select "OpenSatKit" icon with a single click**
  - This launches COSMOS's Command and Telemetry Server, Telemetry Viewer, and displays OSK's main window
  - You can minimize the COSMOS tools, but don't close them

** See COSMOS Appendix for a brief description of each tool

OSK Main Screen

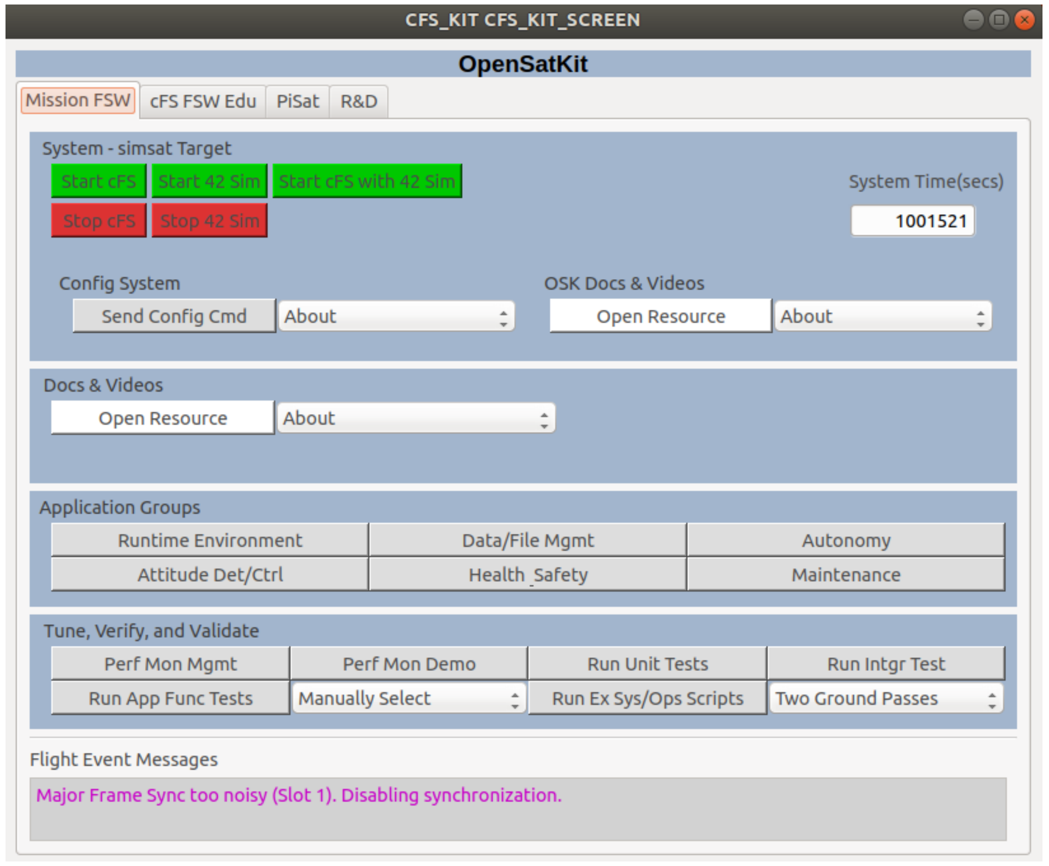
- Four tabs ***Mission FSW***, ***cFS FSW Edu***, ***PiSat***, and ***R&D*** provide the top-level organization
- Each main tab screen has a similar layout

Tabs

System/Target Management

Learning Resources

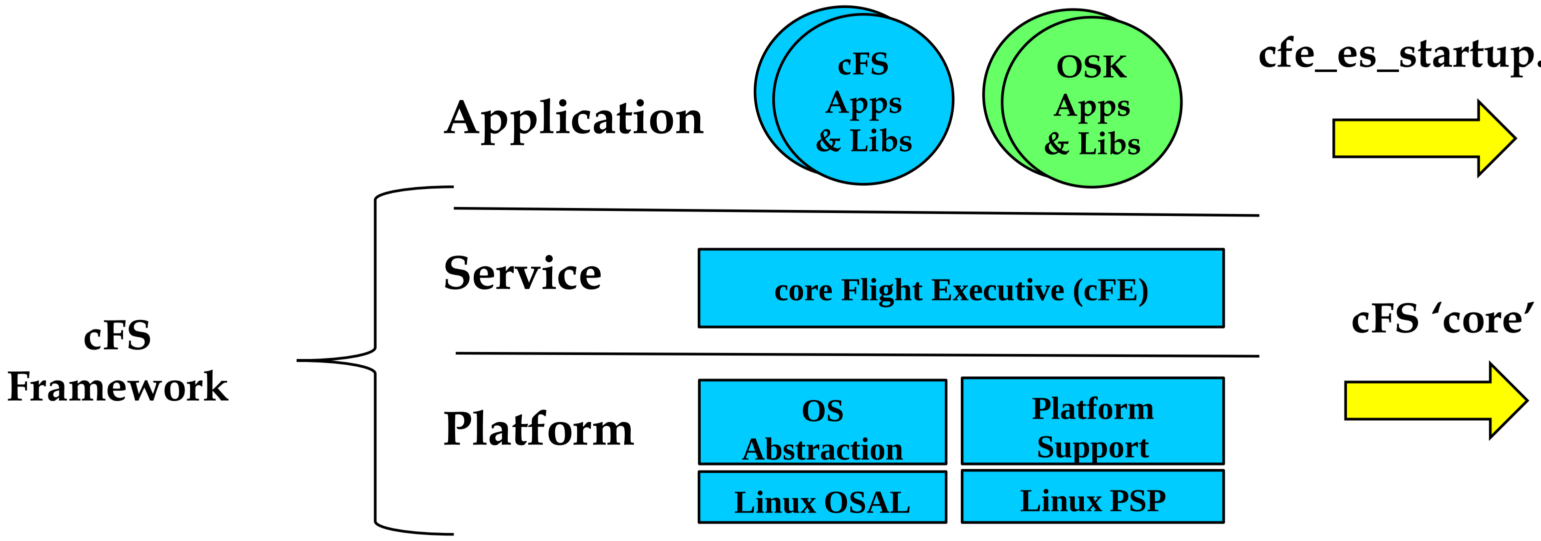
Objective Specific Content



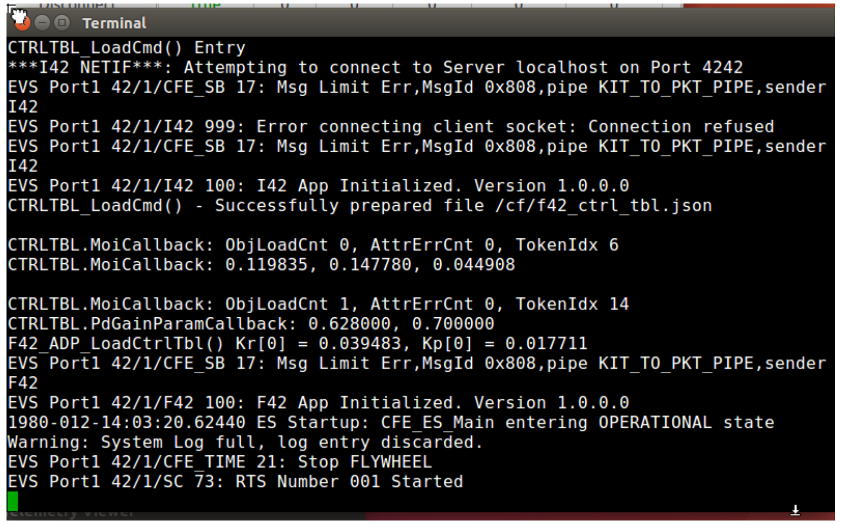
Major Frame Sync too noisy (Slot 1). Disabling synchronization.

Start the Flight Software (FSW)

- **Each main tab screen has a <Start cFS> button to run the cFS target specific for that objective**
  - Targets that include the I42 and F42 apps can optionally be started with the 42 simulator
- **The <Start cFS> button invokes a ruby script that creates a new terminal window executing the "cFS Framework" as a Linux process**
  - When prompted for a password enter your user account password
- **The cFS Framework is the bottom two layers of the 3-tiered cFS architecture. It is a portable application runtime environment that uses a startup script (cfe_es_startup.scr) to determine which apps to load during initialization.**
  - Each target has its own cfe_es_startup.scr



Terminal Window



```
CTRLTBL_LoadCmd() Entry
***I42 NETIF***: Attempting to connect to Server localhost on Port 4242
EVS Port1 42/1/CFE_SB 17: Msg Limit Err,MsgId 0x808,pipe KIT_TO_PKT_PIPE,sender
I42
EVS Port1 42/1/I42 999: Error connecting client socket: Connection refused
EVS Port1 42/1/CFE_SB 17: Msg Limit Err,MsgId 0x808,pipe KIT_TO_PKT_PIPE,sender
I42
EVS Port1 42/1/I42 100: I42 App Initialized. Version 1.0.0.0
CTRLTBL_LoadCmd() - Successfully prepared file /cf/f42_ctrl_tbl.json

CTRLTBL.MoiCallback: ObjLoadCnt 0, AttrErrCnt 0, TokenIdx 6
CTRLTBL.MoiCallback: 0.119835, 0.147780, 0.044908

CTRLTBL.MoiCallback: ObjLoadCnt 1, AttrErrCnt 0, TokenIdx 14
CTRLTBL.PdGainParamCallback: 0.628000, 0.700000
F42_ADP_LoadCtrlTbl() Kr[0] = 0.039483, Kp[0] = 0.017711
EVS Port1 42/1/CFE_SB 17: Msg Limit Err,MsgId 0x808,pipe KIT_TO_PKT_PIPE,sender
F42
EVS Port1 42/1/F42 100: F42 App Initialized. Version 1.0.0.0
1980-012-14:03:20.62440 ES Startup: CFE_ES_Main entering OPERATIONAL state
Warning: System Log full, log entry discarded.
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/SC 73: RTS Number 001 Started
```

FSW Starting Notes

- **If any cFS processes are running you will be prompted to enter a password word so they can be deleted before the new target is started**
  - Targets that include the I42 and F42 apps can optionally be started with the 42 simulator

- **In a few seconds the System Time box should turn white time with advancing**
  - If time doesn't advance check the COSMOS Command and Telemetry Server and make sure "CFS_INT" is connected. If it isn't used the COSMOS buttons to reconnect
  - After the COSMOS server is connected, in the OSK main screen select <Enable Tlm> under "Config System"

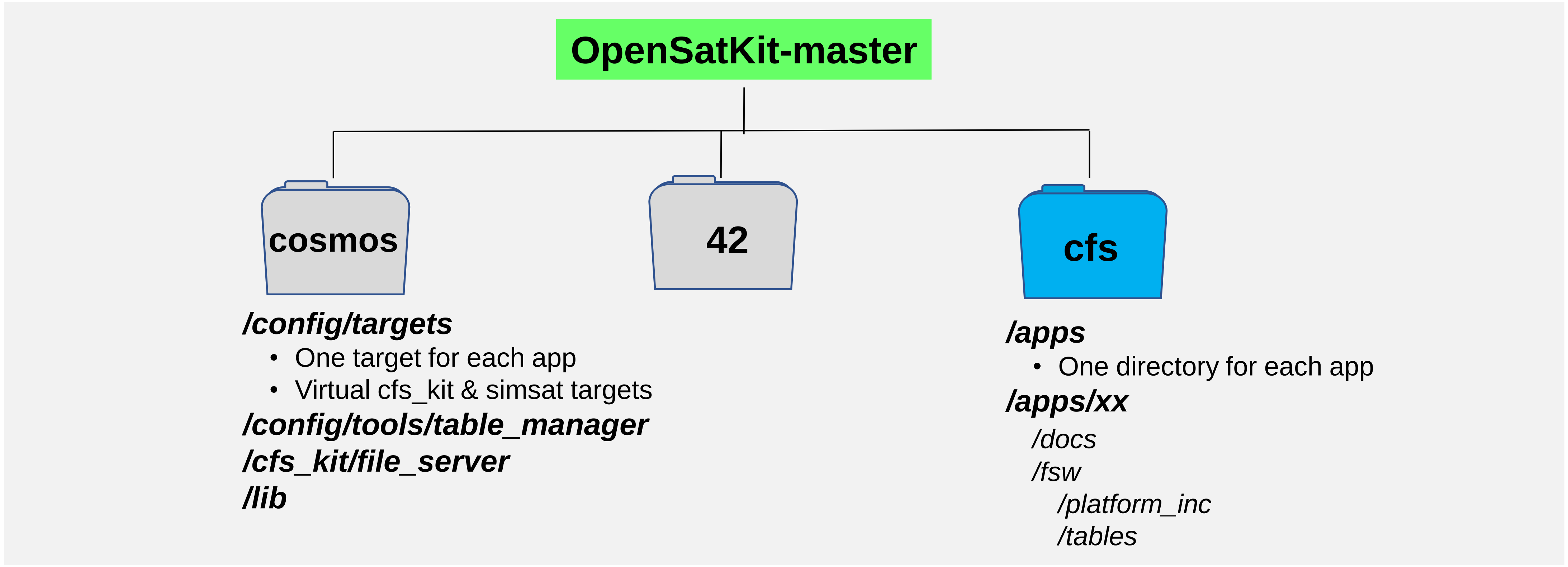
Next Steps

- Separate Quick Start Guides exist for the four user objectives
  - Refer to those guides for details

- The remainder of this guide covers OSK architectural features that apply to all of the user objectives

OSK System Overview

Directory Highlights



- **COSMOS Targets are architectural components that define remote systems that communicate through COSMOS Interfaces**
  - Contain command, telemetry and screen definitions and ruby procedure & library scripts
  - ***cfs_kit*** target defines OSK screens and ruby scripts that have an OSK scope
  - ***simsat*** target defines screens and ruby scripts that are specific to the SimSat reference mission
- **/cosmos/config/tools/table_manager** contains binary file and table definition files
- **/cosmos/cfs_kit/file_server** used for transferring files between ground and flight. “tables” subdirectory used for table transfers
- **/cosmos/lib** defines OSK extensions to COSMOS

COSMOS Configuration (1 of 2)

- **COSMOS *Target* (OpenSatKit/cosmos/config/targets)**
  - Architectural component, typically on an embedded system, that COSMOS can send commands to and receive telemetry from
  - For each target users can define command packets, telemetry packets, screens, and Ruby scripts.
  - Each FSW application is defined as a target
  - OSK defines a virtual target ***CFS_KIT*** to serve as the User's primary interface
  - OSK defines a virtual target ***SIMSAT*** to serve as a reference mission

- **OSK scripts in *OpenSatKit/cosmos/lib* extend COSMOS scripting API**
  - API documentation is under development. See code for details

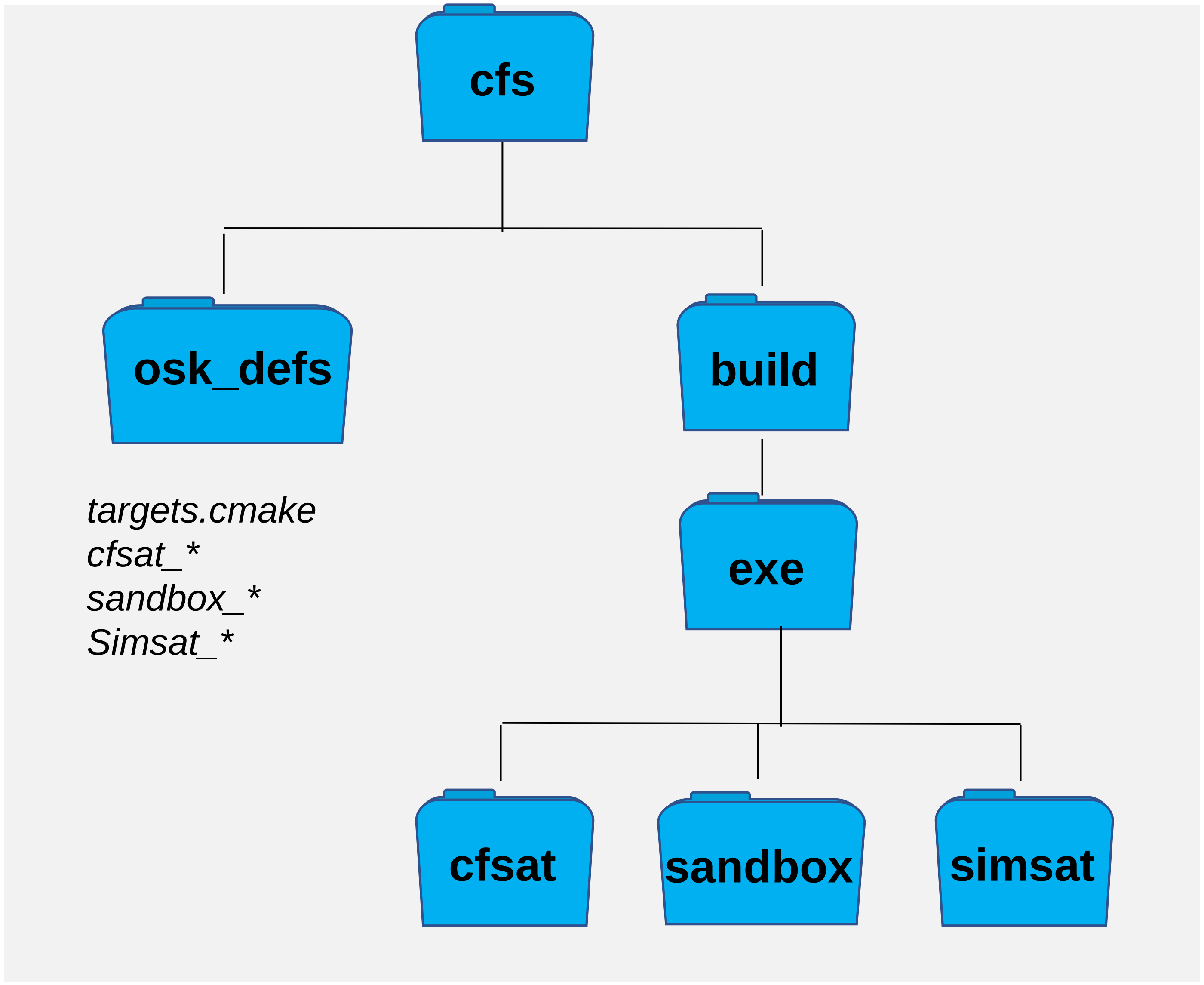
COSMOS Configuration (2 of 2)

- **OSK specific directories defined in *OpenSatKit/cosmos/cfs_kit***
  - */docs*: cFE and OSK documentation
  - */file_server*: Default location for file transferred to/from FSW
    - */table* subdirectory contains table files
    - COSMOS Table Manager file formats defined in */cosmos/config/tools/TableManager*
  - */tools*: cFE and OSK standalone tools
  - */tutorials*: Tutorial files

OSK Conventions

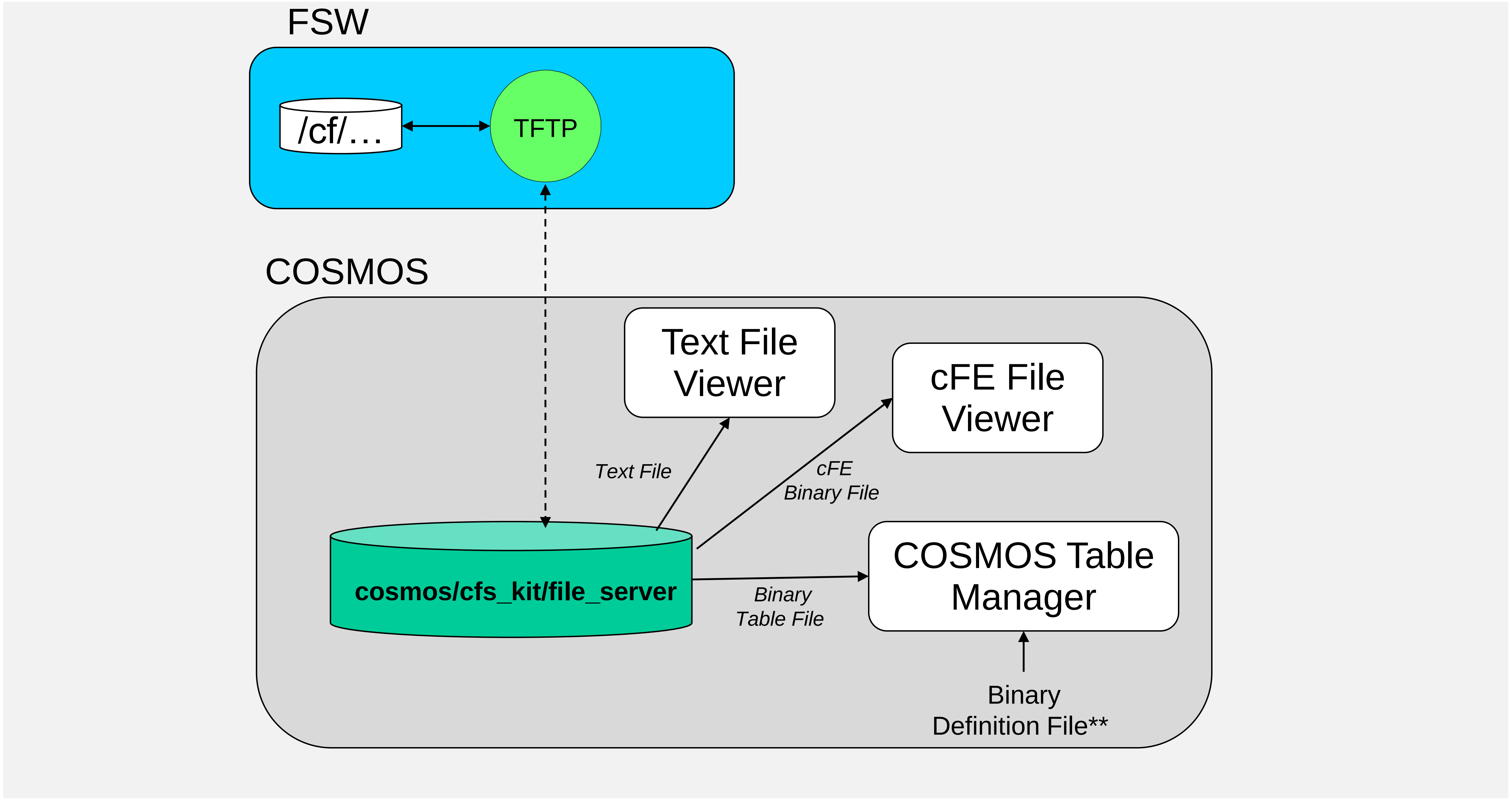
- **Most cFE services have commands that can generate a telemetry as part of the response or write information to a file**
  - The verbs *list* and *send* indicate information is sent in a telemetry packet.
  - *Write* is used when information is written to a file
- **The FSW directory /cf (compact flash) is used as the default location for onboard file creation and flight-ground file transfers**
  - This is mapped to *OpenSatKit/cfs/build/exe/cpu1/cf*
- **OpenSatKit/cosmos/cfs_kit/file_server is used as the default ground file location**
  - Table are located in the *tables* subdirectory
- **OSK often uses osk_tmp_bin.dat as a standard temporary binary file name to avoid clutter**
- **OSK does not "cheat" when working with ground and flight tables**
  - **Files are transferred between flight and ground locations and not accessed via shared locations within the VM**

Flight Software Targets



- **cfsat**
  - cFS FSW Education
- **pisat**
  - Separate repo not shown
  - https://github.com/OpenSatKit/pi-sat
- **sandbox**
  - Research & Development
- **simsat**
  - Mission FSW

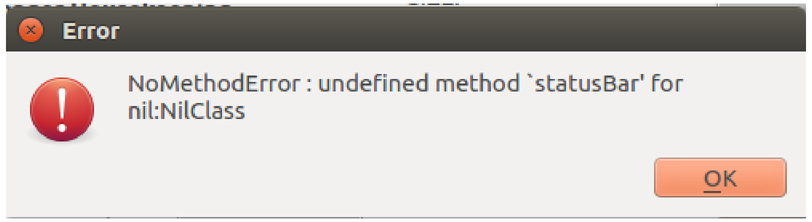
File Management



** Definition files in ~/cosmos/config/tools/table_manager

Minor Inconveniences (1 of 2)

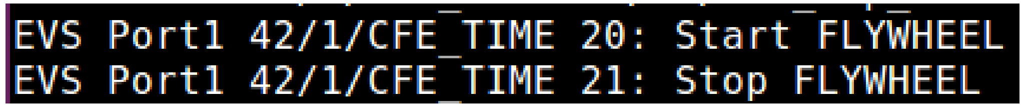
- **OSK is a work in progress with a few known issues that you can ignore**
- **If you cancel an OSK dialogue you may see the follow COSMOS error dialogue.**



Error

NoMethodError : undefined method `statusBar' for nil:NilClass

- **The FSW terminal window may display start and stop "FlyWheel" messages**
  - OSK is a non-realtime environment so the cFE time service is warning that's it's not operating within its real-time precision limits relative to a 1Hz timer
  - OSK is designed to help users learn functional features and only requires reasonable timing performance for the scheduler to execute its schedule correctly



```
EVS Port1 42/1/CFE_TIME 20: Start FLYWHEEL
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
```

Minor Inconveniences (2 of 2)

- Some cFS binary files are variable length. The Table Manager definition files support fixed length files therefore you may see an error dialog stating the file doesn't contain all of the records. This message is from cFE Executive Service Task Information file.

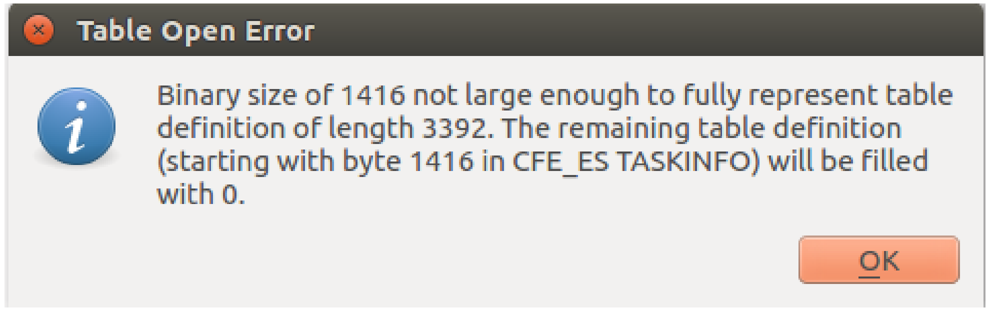


Table Open Error

Binary size of 1416 not large enough to fully represent table definition of length 3392. The remaining table definition (starting with byte 1416 in CFE_ES TASKINFO) will be filled with 0.

4 – COSMOS Appendix

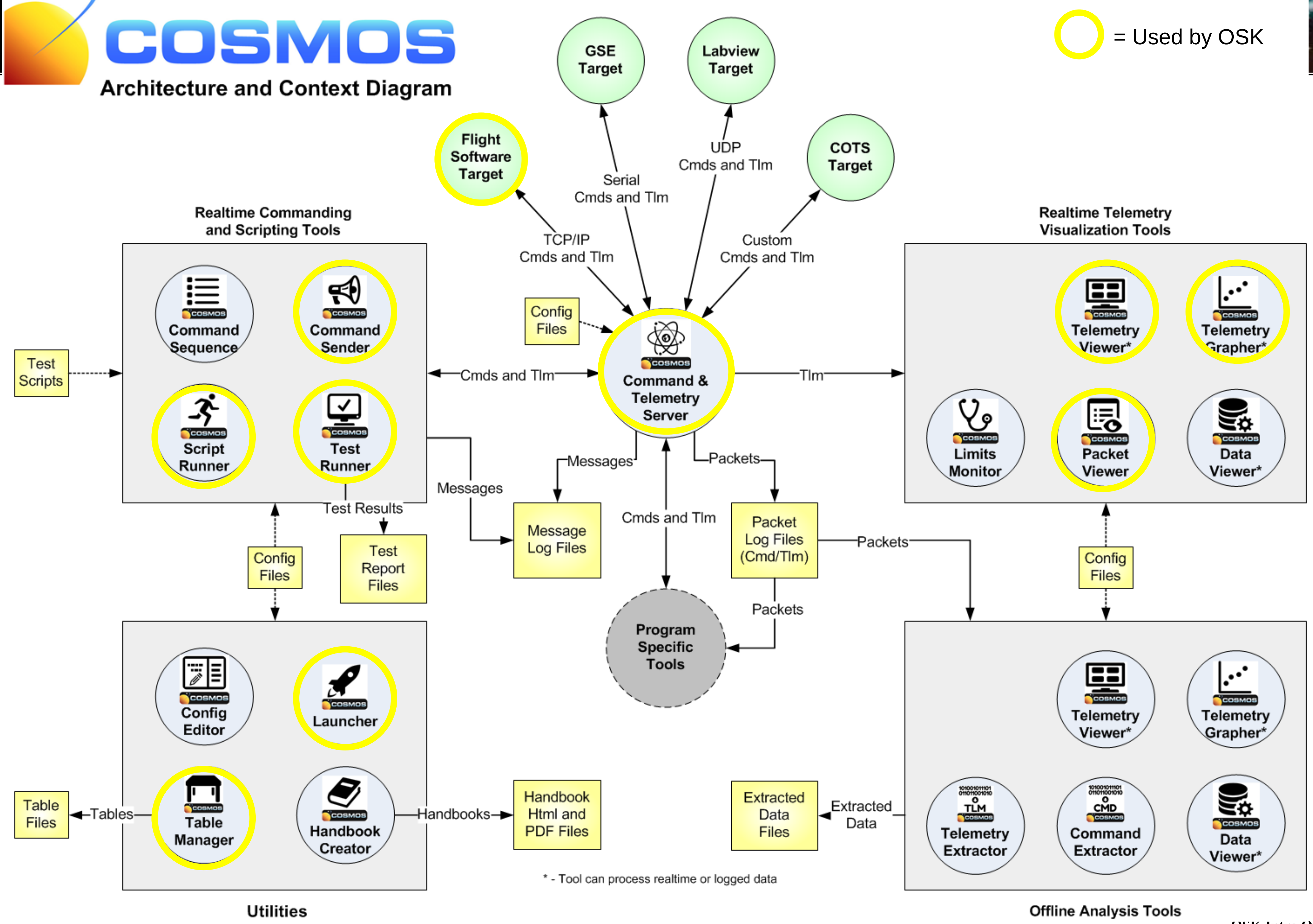
COSMOS Tool Summary (1 of 2)

- **Launcher**
  - Provides a graphical interface for launching each of the tools that make up the COSMOS system
  - *Custom OSK ICON "cFS Starter Kit" launches OSK's main page*
- **Command and Telemetry Server**
  - Connects COSMOS to targets for real-time commanding and telemetry processing.
  - All real-time COSMOS tools communicate with targets through the Command and Telemetry Server ensuring that all communications are logged.
  - Localhost 127.0.0.1 used as cFS connection Targets created
- **Telemetry Viewer**
  - Provides a way to organize telemetry points into custom "screens" that allow for the creation of unique and organized views of telemetry data.
- **Command Sender**
  - Individually send any FSW command using GUI form
  - Raw data files can be used to inject faults
  - *OSK provides custom menus for common cFS commands*
- **Packet Viewer**
  - View any telemetry packet with no extra configuration necessary
  - *OSK provides custom telemetry screens functionally organized*

COSMOS Tool Summary (2 of 2)

- **Telemetry Grapher**
  - Real-time or offline graphing of any FSW telemetry point
  - *OSK provides convenient access through some of its custom screens*
- **Table Manager**
  - Edit and display binary files
  - *OSK provides definitions for most of the cFE binary files and a limited number of cFS application binary files*
- **Script Runner**
  - Develop and execute test procedures using Ruby Scripts and COSMOS APIs
  - *OSK provides additional APIs for functions like file transfer and binary file management*
- **Test Runner**
  - Test framework for organizing, executing, and verifying test scripts
  - *Currently OSK only includes some prototype scripts. The goal is to provide a complete test suite that can be extended by the user.*

COSMOS Architecture and Context Diagram



○ = Used by OSK
Flight Software Target
GSE Target
Labview Target
COTS Target
TCP/IP Cmds and Tlm
Serial Cmds and Tlm
UDP Cmds and Tlm
Custom Cmds and Tlm
Realtime Commanding and Scripting Tools
Command Sequence
Command Sender
Script Runner
Test Runner
Test Scripts
Config Files
Command & Telemetry Server
Cmds and Tlm
Tlm
Realtime Telemetry Visualization Tools
Telemetry Viewer*
Telemetry Grapher*
Limits Monitor
Packet Viewer
Data Viewer*
Messages
Packets
Test Results
Message Log Files
Packet Log Files (Cmd/Tlm)
Test Report Files
Program Specific Tools
Utilities
Config Editor
Launcher
Table Manager
Handbook Creator
Table Files
Tables
Handbooks
Handbook Html and PDF Files
Extracted Data Files
Extracted Data
Offline Analysis Tools
Telemetry Extractor
Command Extractor
* - Tool can process realtime or logged data