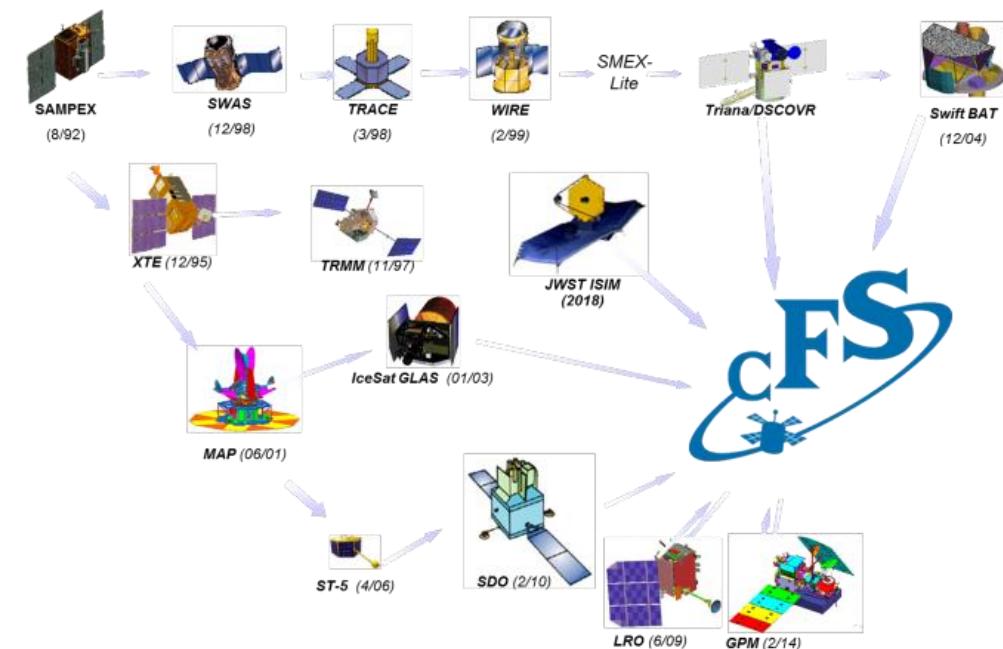




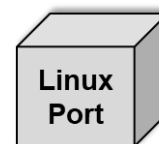
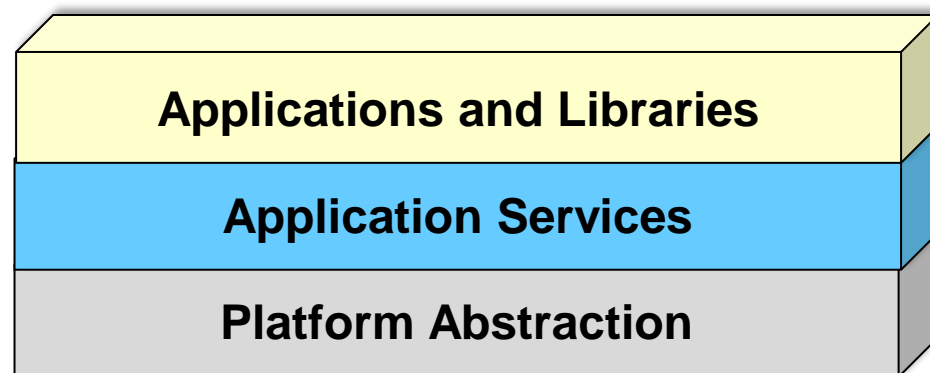
Getting Started with OpenSatKit

OSK v3.3

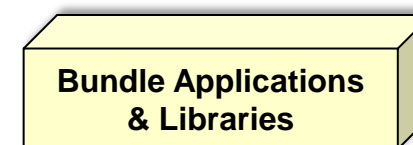


NASA's core Flight System (cFS) is an open-source flight software (FSW) framework providing an application runtime environment that is portable across different hardware/operating system platforms.

The cFS provides a high quality FSW with decades of flight heritage.



**NASA
cFS Framework**



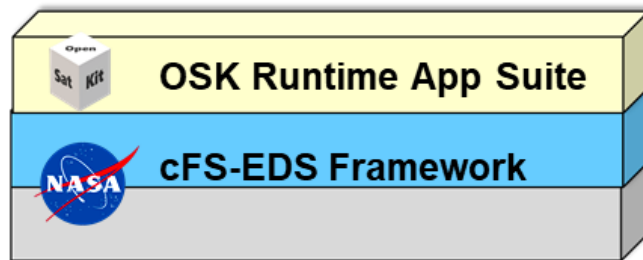
**NASA
cFS Bundle**



OpenSatKit Overview



- **OpenSatKit (OSK) is suite of open-source products built upon NASA's core Flight System (cFS)**
- **OSK products provide**
 1. A cFS educational platform
 2. Mission cFS-based FSW application design & development environment
 3. STEM education and hobbyist platforms
 4. Technology development platforms

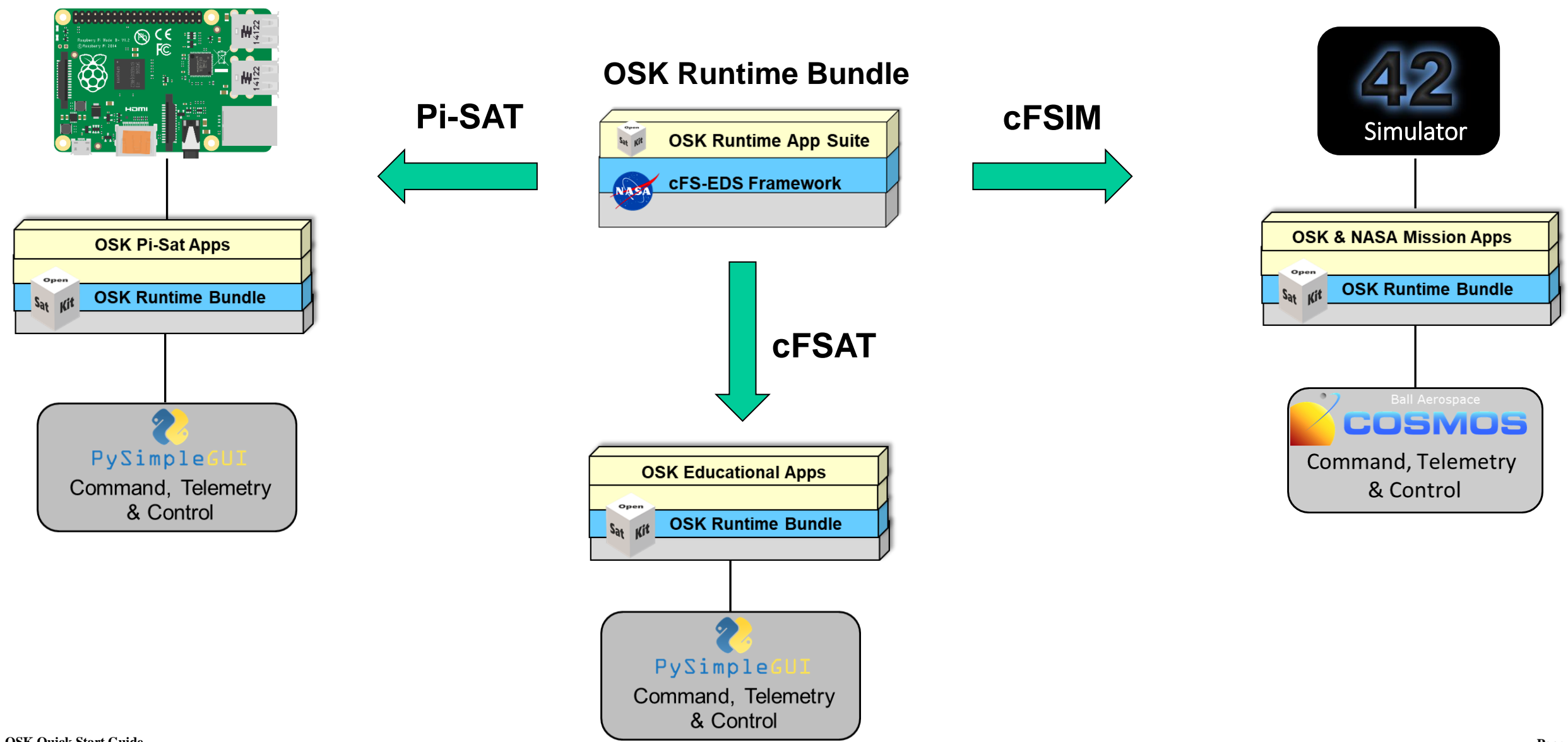


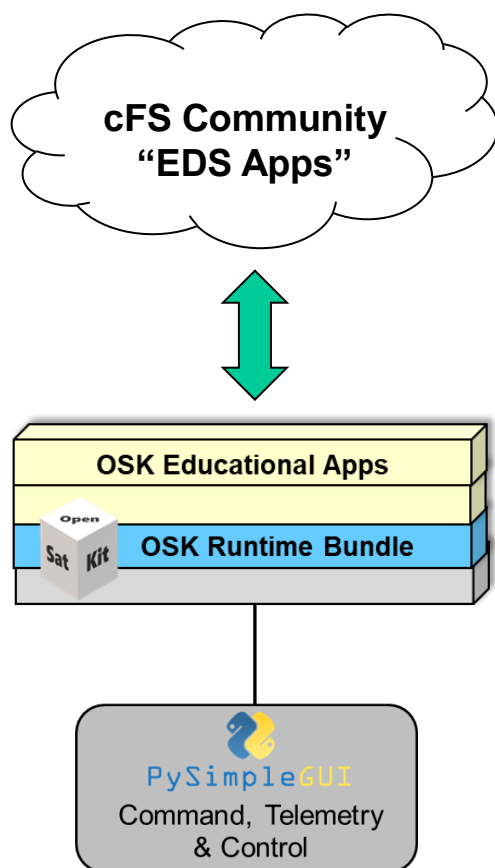
OSK Runtime Bundle

- Built upon NASA's cFS Electronic Data Sheet (EDS) Framework
- OSK app suite provides FSW functionality required by every cFS distribution

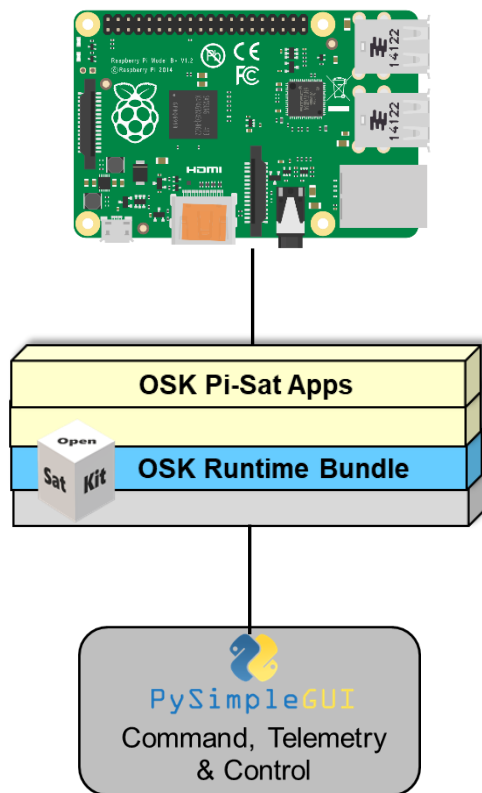


OpenSatKit Product Suite

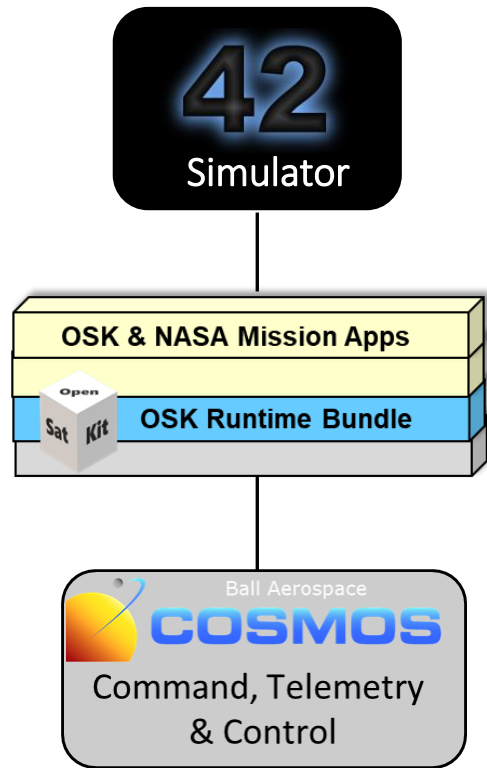




- **Lightweight platform that serves as a cFS**
 - Educational platform
 - Application development environment
- **Suitable for software STEM educational projects including**
 - FSW application development and test
 - Ground software data processing
- **Uses the [cFS with Electronic Data Sheets \(EDS\) Framework](#)**
 - Includes an EDS toolchain that creates ground and FSW artifacts from cFS apps packaged with EDS interface specifications
 - Uses latest cFS Caelum release
- **Includes OSK Runtime App Suite**
 - Command Ingest, Telemetry Output, Scheduler, File Manager, File Transfer
 - Assured compatibility with the cFS Framework
- **Minimalistic Command, Telemetry and Control ground system**
 - Written in a lightweight portable python GUI framework called [PySimpleGUI](#)
- **[OpenSatKit-Apps](#) is a github repository of cFS Apps with EDS specifications**

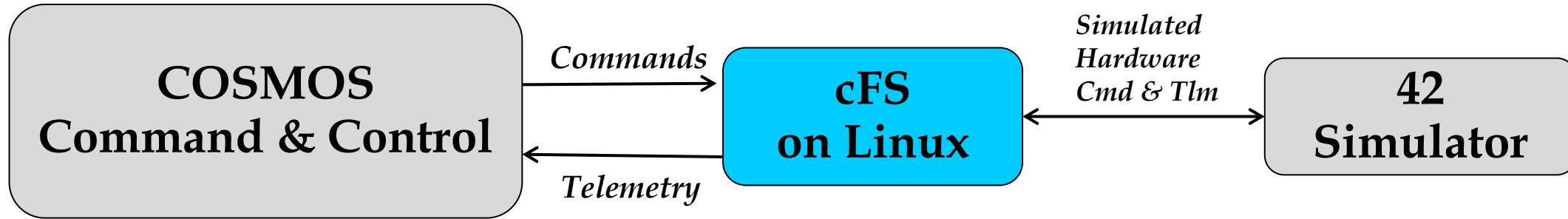


- **Augments cFSAT with Raspberry Pi libraries and apps**
 - Assured compatibility with the cFS Framework
- **Low-cost hardware platform ideal for learning how to write cFS “interface apps”**
 - Provide command and telemetry interface to a hardware device
- **Base platform that can easily be customized for STEM educational projects**



- **Complete ground, flight, and dynamic simulation system including**
 - Ball Aerospace's COSMOS command and control platform for embedded systems
 - NASA Goddard's 42 dynamic simulator
- **Suitable for**
 - FSW application development and test
 - Ground software data processing

- **OSK v3.4 components are being refactored and distributed into separate products**
- **Motivation**
 - Single monolithic ‘one size fits all’ product too cumbersome for many users and hard to maintain
 - Significant cFS Framework changes from release 6.7 and Caelum
 - Significant COSMOS architectural changes from release 4.x to 5.0 and 4.x support duration is unknown
 - Maturation of cFS EDS framework that supports better app packaging and distribution
 - Modular components more suitable for open-source community updates
- **Plan**
 1. Create separate OSK App Repos, <https://github.com/OpenSatKit-Apps>
 - Update apps to cFS Caelum and EDS specifications
 2. Create cFSAT and Pi-Sat products
 - Create ground GUI and tools that support the OSK runtime app suite
 - Migrate cFS educational material
 3. Create cFSIM
 - Transition from COSMOS 4.x to 5.x
 - Migrate cFS mission educational material
- **Challenges**
 - Unknown date for NASA Caelum compliant apps
 - COSMOS 5.0





- **In addition to the cFS, OSK uses two additional open-source applications**
 - Ball Aerospace’s COSMOS command and control platform for embedded systems
 - NASA Goddard’s 42 dynamic simulator
- **OSK’s cFS Distribution**
 - Includes a large app suite composed of cFS community and OSK custom apps
- **Apps are combined into multiple cFS Targets aligned with the four OSK user objective use cases**




OSK Document Roadmap



OpenSatKit/docs

- 1  – *OSK Quick Start*: Top-level introduction to OSK and a roadmap for more in-depth engagement
- 3  – *OSK COSMOS Guide*: Describes how COSMOS has been configured and extended for OSK
- *OSK App Developer's Guide*: Describes how to develop apps using the OSK application framework

OpenSatKit/cosmos/config/targets/CFSAT/docs (cFS educational platform)

- 2  {
 - *cFS Education Quick Start Guide*: Introduction to OSK's cFS educational target and associated resources
 - *core Flight System (cFS) Overview*: Introduction to flight software (FSW) and NASA's cFS
 - *core Flight Executive (cFE) Overview*: Overview of the cFE framework and its application services

OpenSatKit/cosmos/config/targets/CFE_[service] /docs

- Each cFE service contains its own tutorial document

OpenSatKit/cosmos/config/targets/SIMSAT/docs (cFS-based mission)

- *Mission FSW Quick Start Guide*: Introduction to OSK's cFS-based mission target and associated resources
- *Simple Sat Overview*: Describes the SimpleSat reference mission
- *Application Group Guides*: Multiple documents that describe how groups of cFS community apps work together

OpenSatKit/cosmos/config/targets/PISAT/docs (Raspberry Pi distro for STEM education)

- *Pi-Sat Quick Start Guide*: Introduction to OSK's Raspberry Pi target and associated resources

OpenSatKit/cosmos/config/targets/SANDBOX/docs (cFS application playground)

- *Research & Development Quick Start Guide*: Introduction to OSK's R&D target and associated resources



Recommended reading order if you're new to the cFS. The next steps depends on your goals.

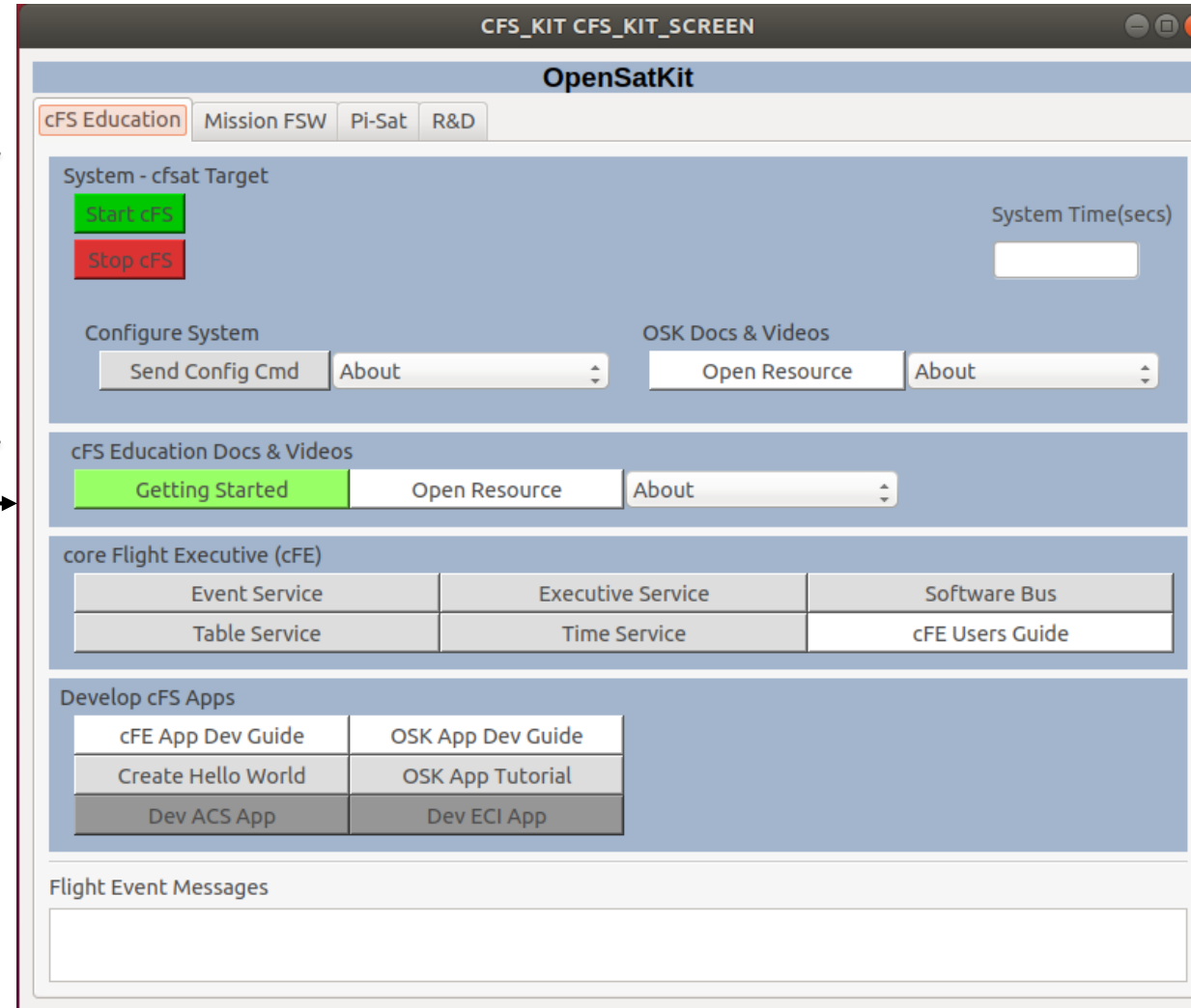
- Four tabs **cFS Education**, **Mission FSW**, **PiSat**, and **R&D** correspond to four cFS targets/ OSK Use Cases
- Each tab's screen has a similar layout and its own "Getting Started" Guide

User Objective Tab

System/Target Management

Learning Resources

Use Case Specific Content





OSK YouTube Channel



OSK Wiki

OpenSatKit Training Videos

Introduction

- OSK Purpose
- OSK Installation
- cFS Overview
- cFS Build Overview

Learn the cFS Framework

- cFE Executive Service coming soon...
- cFE Event Service
- cFE Software Bus
- cFE Table Service
- cFE Time Service coming soon...

Systems Engineering

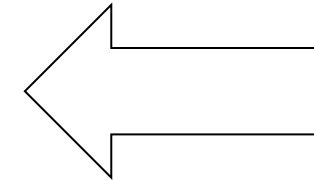
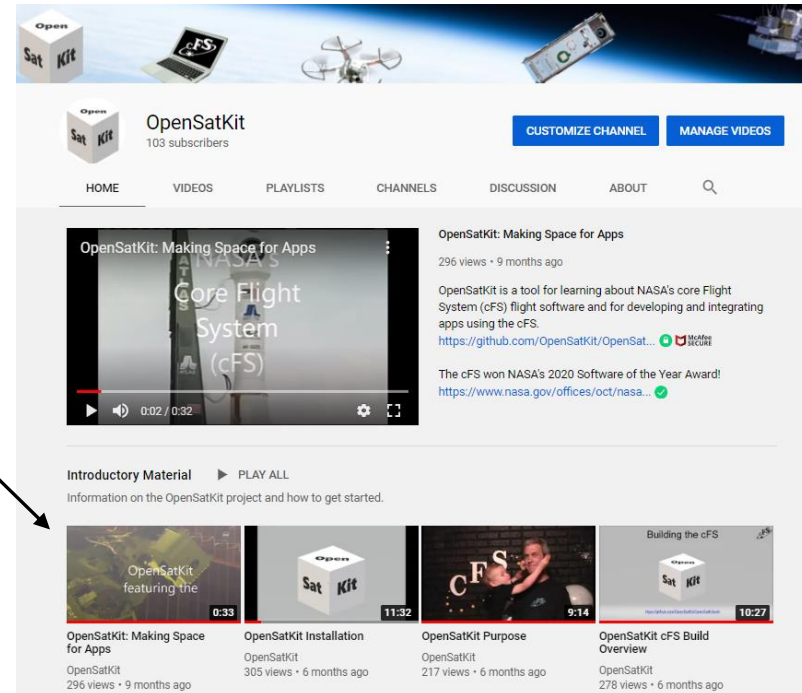
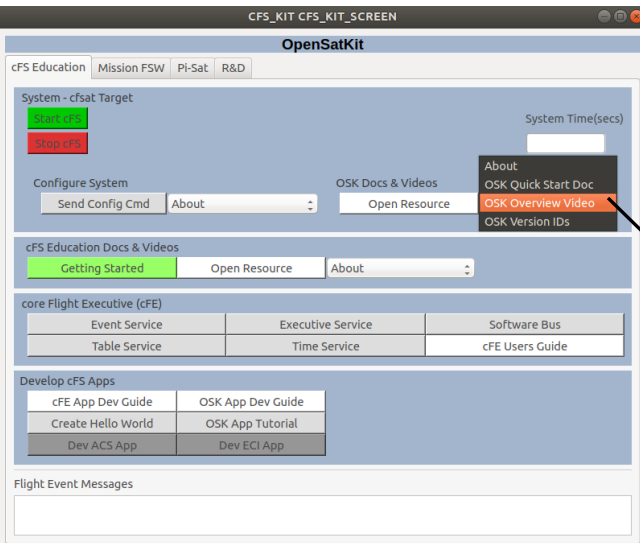
- Runtime Environment: CI, SCH, TO
- Data-File Management Intro: CF, DS, FM, HK, TFTP
- Health&Safety App Group Intro: CS, HS
- Maint App Group Intro: MD, MM
- Performance Monitor

Learn Community Apps

- OSK App Ops Intro
- File Manager App
- Housekeeping App

Develop Apps

- Create "Hello World" App
- "Hello World" App Code Inspection



- OSK and cFS training videos organized into playlist
- Convenient video access from OSK screens and OSK wiki

<https://www.youtube.com/channel/UC2wfvAlkrrgyC4ITwL3zokg/>

- 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
 - Run 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

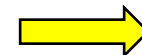
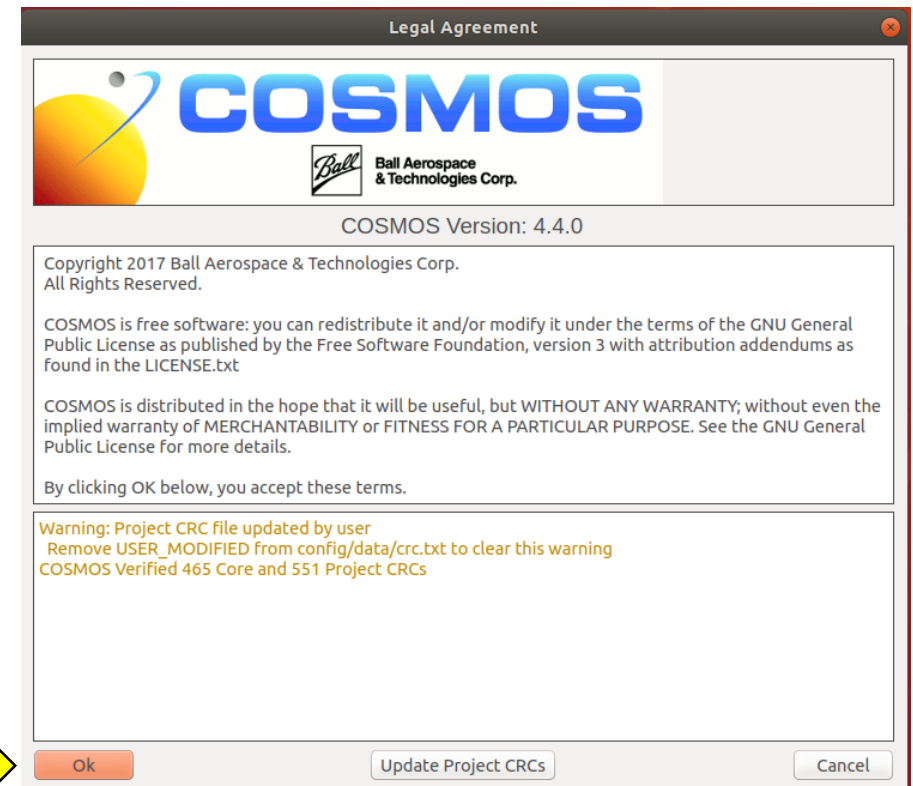
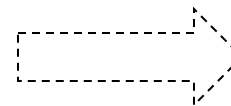
**If you're new to the cFS start with the cFS
Education tab and read the Getting Started Guide**



Running OSK

1. Open a terminal window (Ctrl-Alt-t)
2. 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/cfs” if OpenSatKit was installed in the home directory for an account named “user”
3. Change directory to cosmos
 - cd ~/cosmos
4. Start COSMOS
 - ~/cosmos\$ ruby Launcher
5. Select <OK> to create the “Launcher” screen shown on the next slide

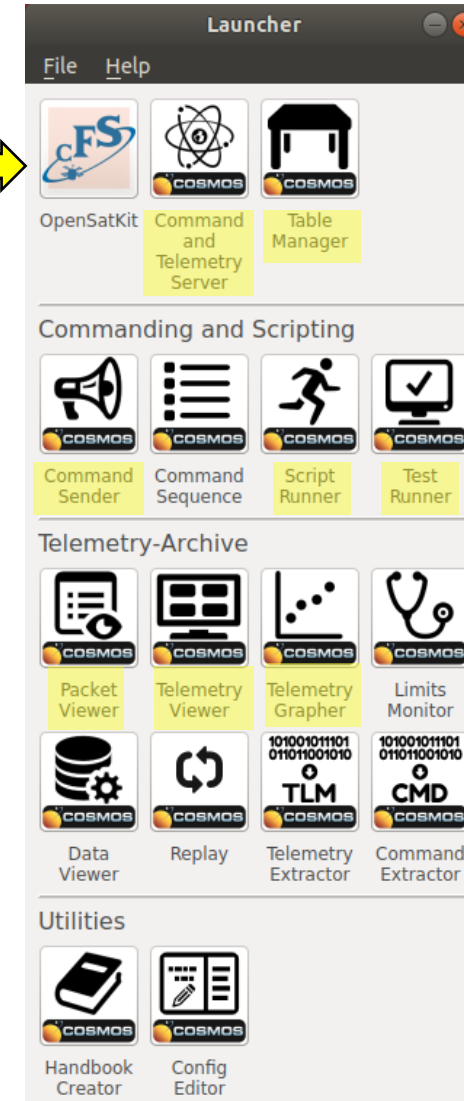
```
osk@ubuntu: ~/OpenSatKit/cosmos
File Edit View Search Terminal Help
osk@ubuntu:~$ cd OpenSatKit/cosmos
osk@ubuntu:~/OpenSatKit/cosmos$ ruby Launcher
```



6. Select “OpenSatKit” icon with a single click

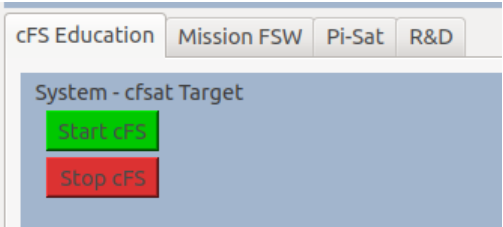
- This launches COSMOS’s Command and Telemetry Server, Telemetry Viewer, and displays OSK’s main window
- You can minimize the COSMOS tools, but don’t close them

- 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

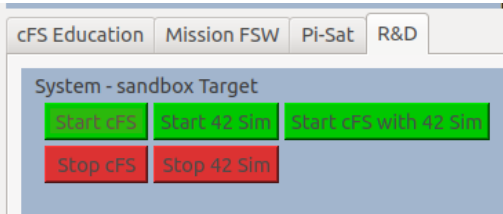
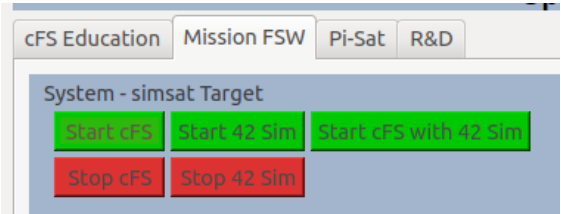


** See COSMOS Appendix for a brief description of each tool

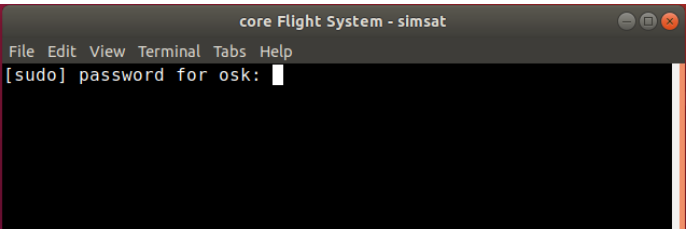
- The **cFS Education**, **Mission FSW**, and **R&D** tabs each have **System** section at the top of their screen
 - See the *PiSat Getting Started Guide* for starting a Pi-Sat target
- The **cFS Education** target **cfsat** does not include the 42 simulator apps so it can't be run with the 42 simulator



- The **Mission FSW (simstat)** and **R&D (sandbox)** targets do include the 42 simulator apps so they can optionally be run with the 42 simulator



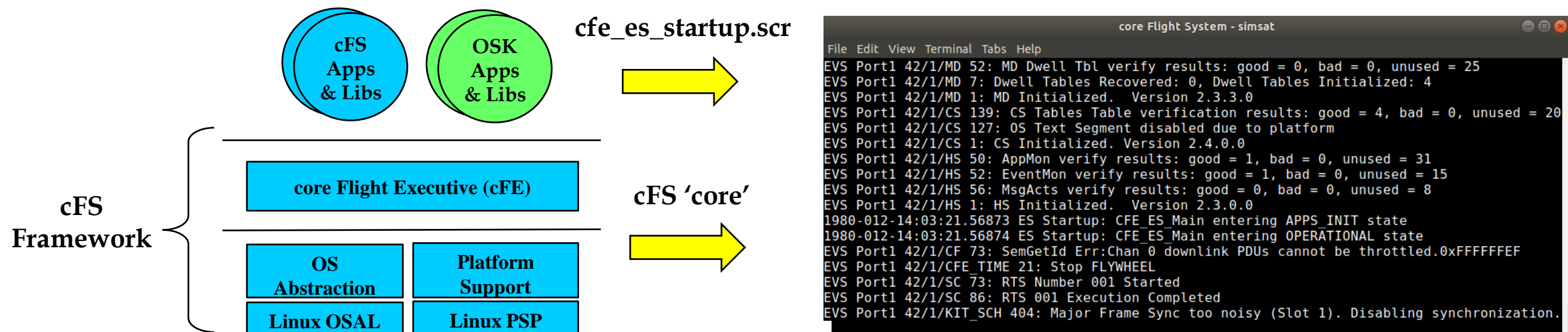
- The **Start cFS** button create a terminal window with target name in the title



When prompted enter your user account password. This will run the cFS Framework as a Linux process.

- 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



The cFS is designed for an embedded processor environment so the messages in the terminal window are the same messages that would appear if a terminal screen were attached to a UART port on an embedded processor card



Starting a cFS Target 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 with time advancing**
 - If time doesn't advance check the COSMOS Command and Telemetry Server and make sure "CFS_INT" is connected. If it isn't use, the COSMOS buttons to reconnect
 - After the COSMOS server is connected, in the OSK main screen select <Enable Tlm> under "Config System"



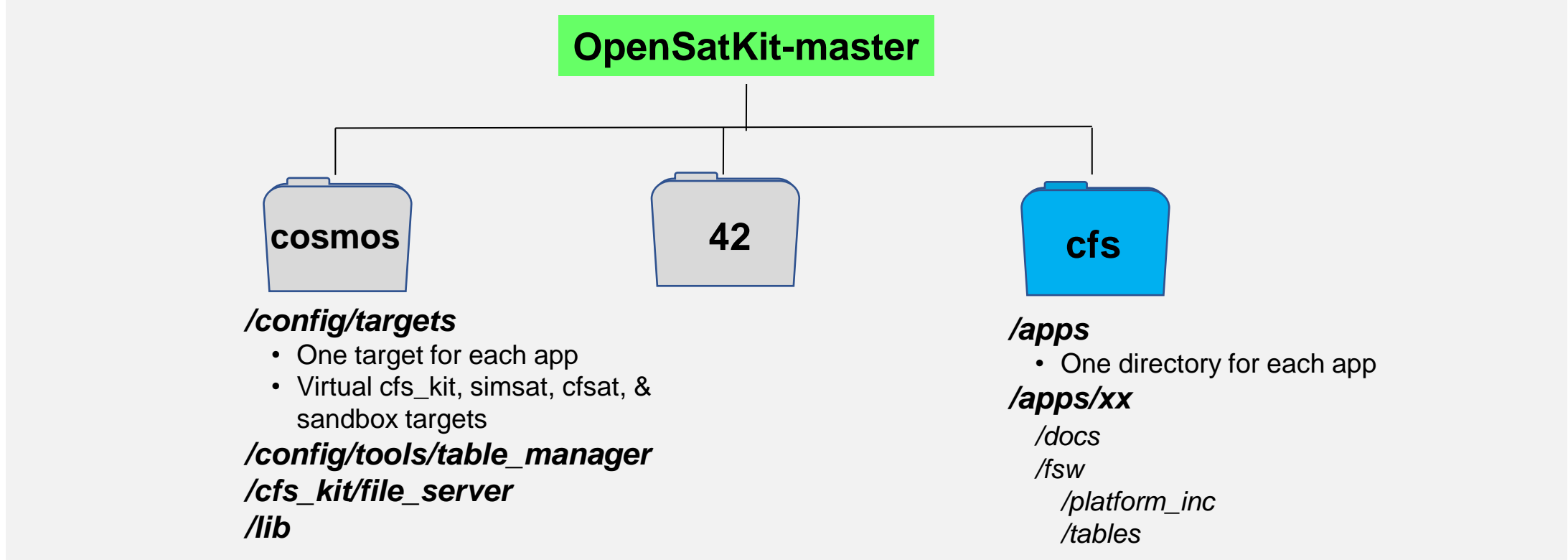
Next Steps



- **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 the user objectives**



OSK System Overview



- **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 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

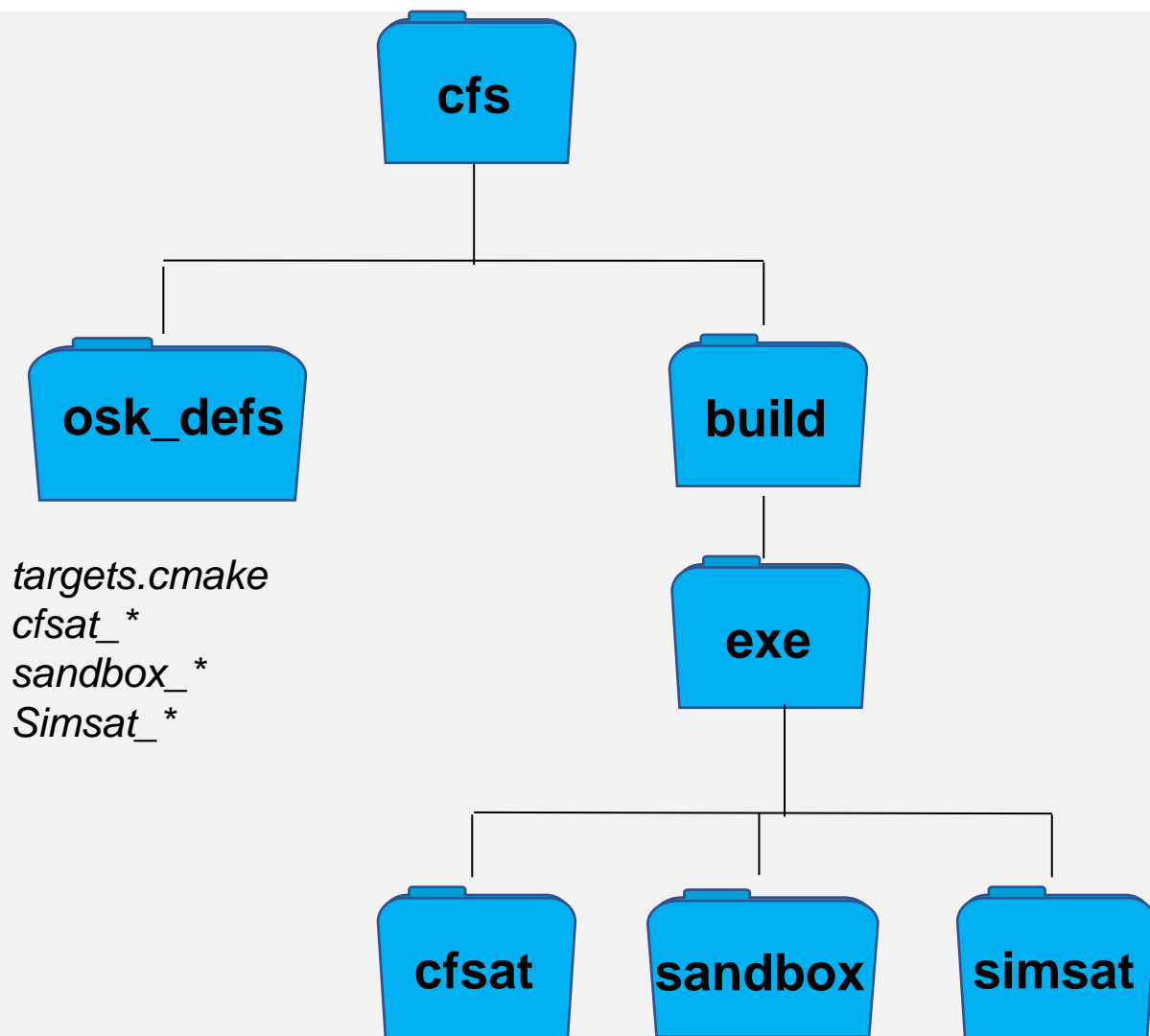


COSMOS Configuration (2 of 2)

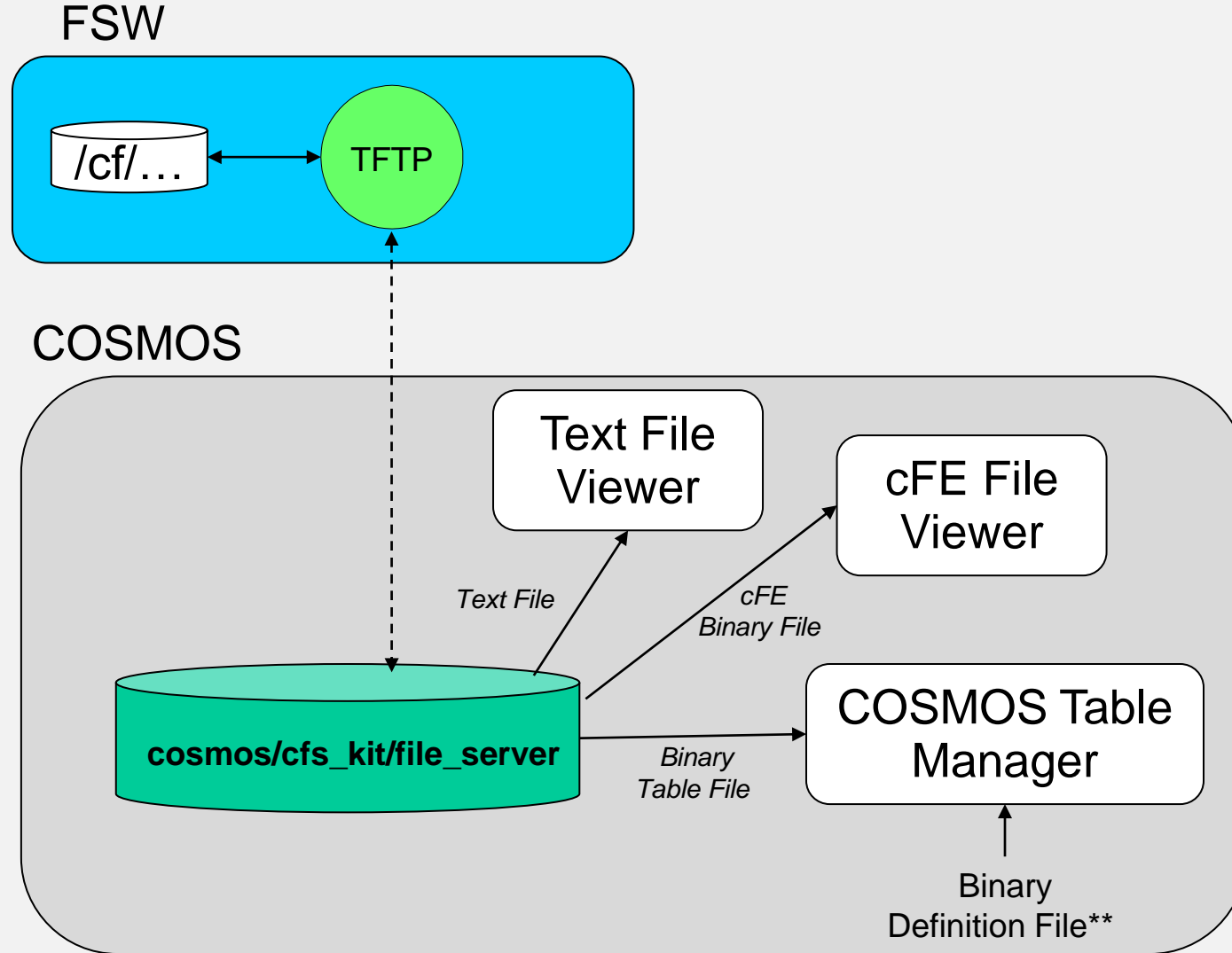


- **OSK specific directories defined in *OpenSatKit/cosmos/cfs_kit***
 - */docs*: cFE and OSK documentation
 - */file_server*: Default location for file transferred to/ from FSW
 - */table* subdirectory contains table files
 - COSMOS Table Manager file formats defined in */cosmos/config/tools/TableManager*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

- 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

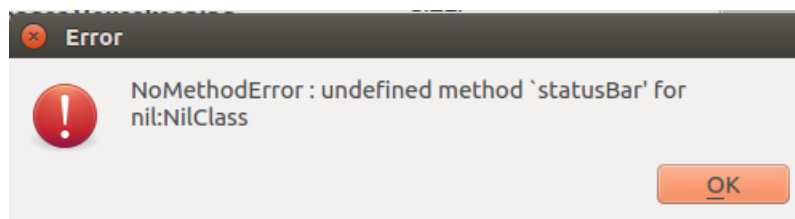


- **cfsat**
 - cFS FSW Education
- **pi-sat**
 - Separate repo not shown
 - <https://github.com/OpenSatKit/pi-sat>
- **sandbox**
 - Research & Development
- **simsat**
 - Mission FSW



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

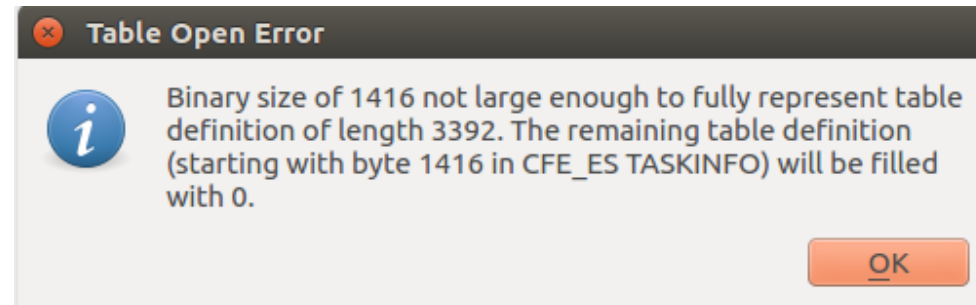
- OSK is a work in progress with a few known issues that you can ignore
- If you cancel an OSK dialogue you may see the follow COSMOS error dialogue.



- 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
```

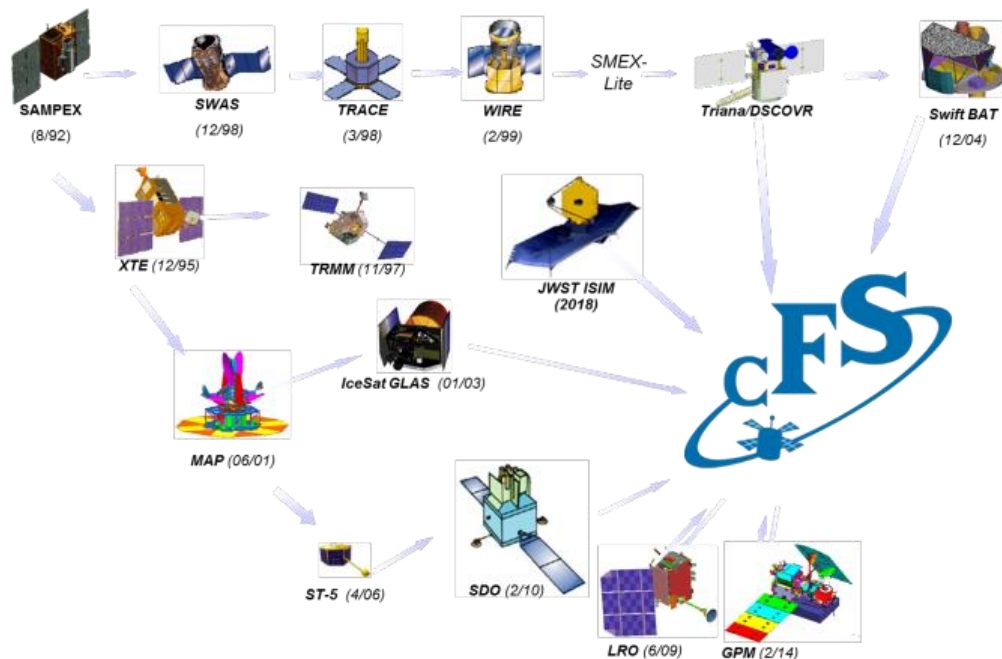

- Some cFS binary files are variable length. The Table Manager definition files support fixed length files therefore you may see an error dialog stating the file doesn't contain all of the records. This message is from cFE Executive Service Task Information file.





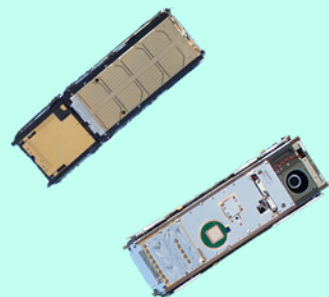
Appendix A

cFS Extras



cFS developed by NASA/Goddard to provide a FSW platform to support diverse custom architectures for near-earth scientific spacecraft

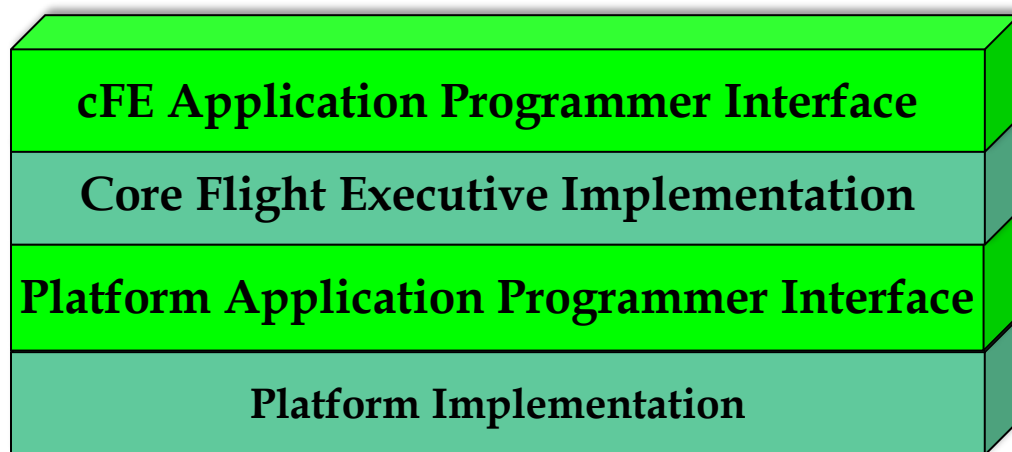
- Extensive flight heritage and highly reliable
- Portable across heterogenous platforms
- Application-centric design that supports application reuse
- Highly configurable with compile-time and runtime parameters
- Maintainability (diagnostics and repair) built into services and reusable app suite



Large user community with missions ranging from CubeSats to NASA's Lunar Gateway

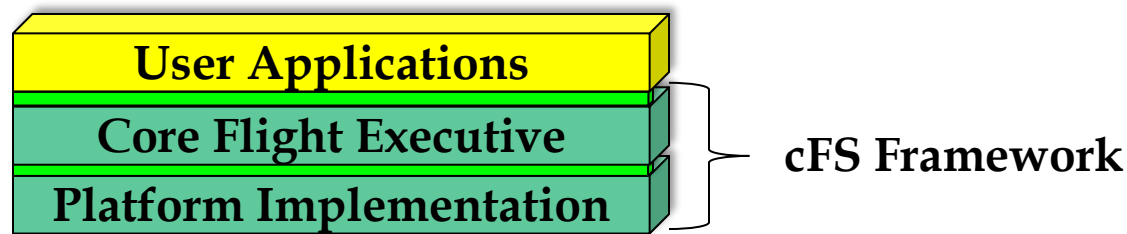


- A NASA multi-center configuration controlled open-source flight software framework

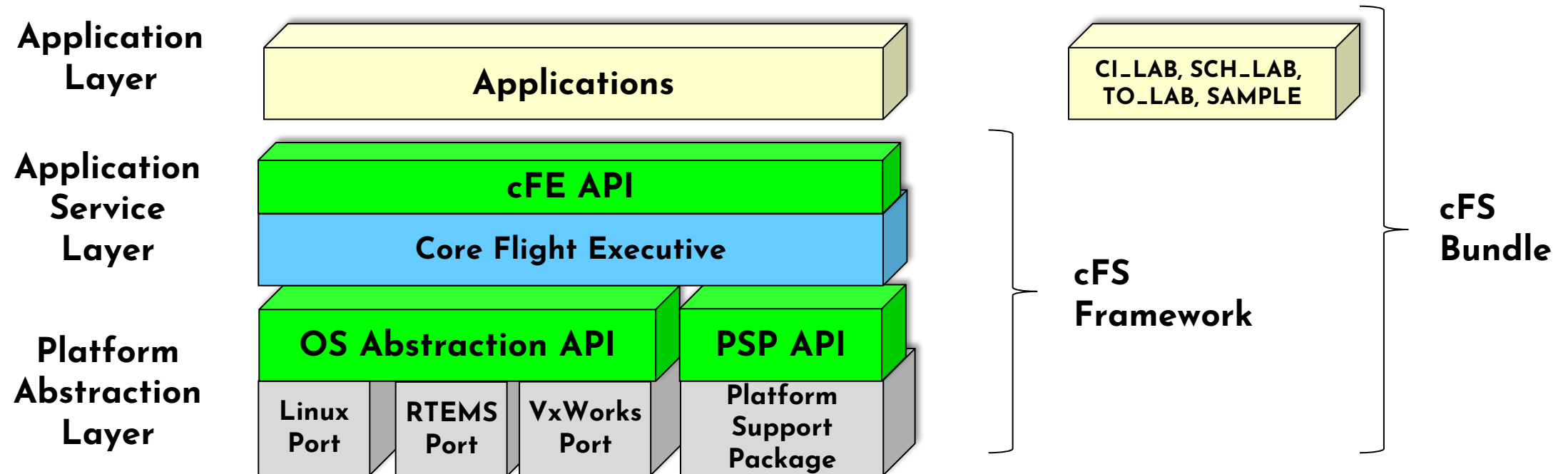


- Layered architecture with international standards-based interfaces
- Provides development tools and runtime environment for user applications
- Reusable NASA Class A/B lifecycle artifacts: requirements, design, code, tests, and documents

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

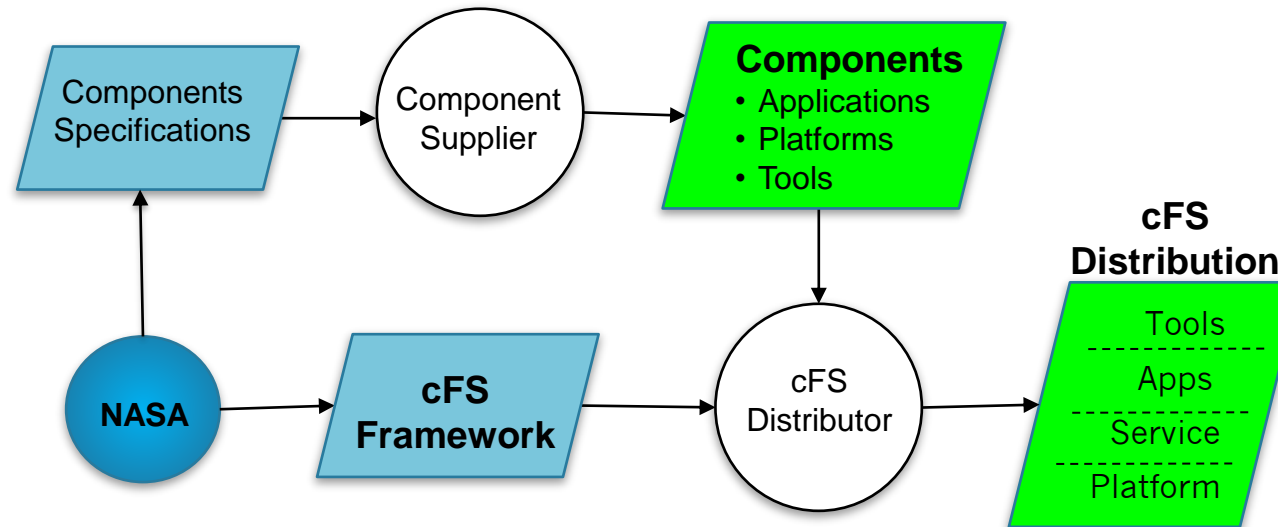


- A worldwide community from government, industry, and academia



NASA cFS Governance

- A NASA multi-center configuration control board maintains the cFS Framework and Bundle apps
 - All released under Apache 2.0
 - User contributions accepted
- Applications are maintained by the organization that developed it
 - NASA apps are either NASA Open-Source Agreement (NOSA) or Apache 2.0



- 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



Appendix B

COSMOS Extras



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*

- **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.*

