NASA/TM–20205000691

Core Flight System (cFS)Training

Flight Software Systems Branch, Code 582
Goddard Space Flight Center, Greenbelt, MD

April 2020

NASA STI Program ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

Write to:
NASA STI Information Desk
Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199

NASA/TM–20205000691

Core Flight System (cFS)Training

Flight Software Systems Branch, Code 582
Goddard Space Flight Center, Greenbelt, MD

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, MD 20771

April 2020

Notice for Copyrighted Information

This manuscript is a work of the United States Government authored as part of the official duties of employee(s) of the National Aeronautics and Space Administration. No copyright is claimed by the United States under Title 17, U.S. Code. All other rights are reserved by the United States Government. Any publisher accepting this manuscript for publication acknowledges that the United States Government retains a non-exclusive, irrevocable, worldwide license to prepare derivative works, publish, or reproduce this manuscript, or allow others to do so, for United States Government purposes.

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA STI Program
Mail Stop 148
NASA's Langley Research Center
Hampton, VA 23681-2199

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
703-605-6000

Available in electronic form at https://www.sti.nasa.gov and https://ntrs.nasa.gov

National Aeronautics and Space Administration

Core Flight System (cFS) Training

Module 1: Introduction

Course Agenda

1. Introduction
2. cFE Services
    a) Executive Services
    b) Time Services
    c) Event Services
    d) Software Bus
    e) Table Services
3. Application Layer
    a) cFS Applications
    b) cFS Libraries

Course Audience & Prerequisites

- **NASA Flight Software Developers**
- **Prerequisites:**
  - C programming experience
  - Linux experience

- **System requirements for hands-on exercises:**
  - Linux build environment
    - With sudo privileges or a /proc/sys/fs/mqueue/msg_max >= 1024
  - git, gcc, cmake, clang
  - Python, PyQt4, PyZMQ

Course Learning Objectives

- Understand the architecture of the cFS
- Build and execute the cFS
- Interact with the cFS through a ground system
- Add an app to a cFS system

Introduction Agenda

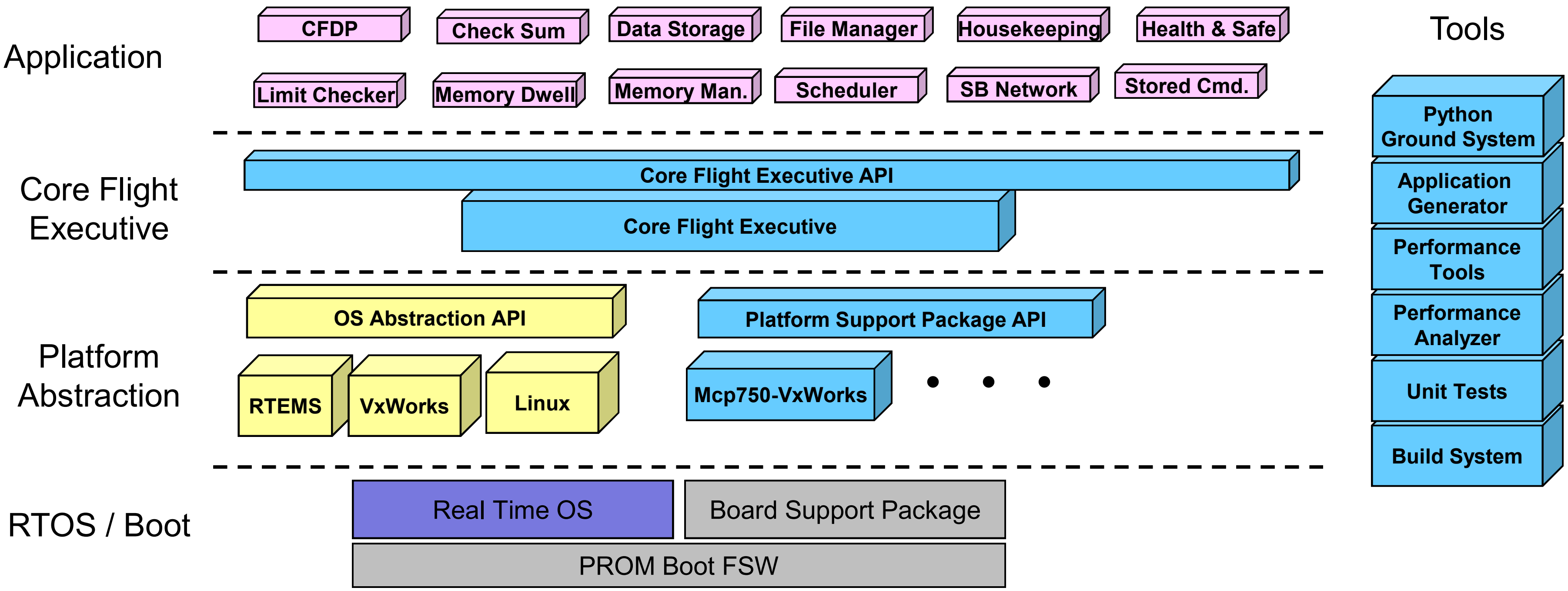
- What is cFS?
- cFS Community
- cFS Architectural Overview

What is cFS?

cFS Overview

- **A platform and project independent reusable software framework and set of reusable software applications**
  - Platform Abstraction Layer supports portability
  - Applications provide mission functionality
  - Compile-time configuration parameters and run-time command/table parameters add flexibility and scalability
- **Key aspects:**
  - Dynamic run-time environment
  - Layered architecture
  - Component-based design

cFS Architecture Layers



Application

CFDP
Check Sum
Data Storage
File Manager
Housekeeping
Health & Safe
Limit Checker
Memory Dwell
Memory Man.
Scheduler
SB Network
Stored Cmd.

Core Flight Executive

Core Flight Executive API
Core Flight Executive

Platform Abstraction

OS Abstraction API
RTEMS
VxWorks
Linux
Platform Support Package API
Mcp750-VxWorks
• • •

RTOS / Boot

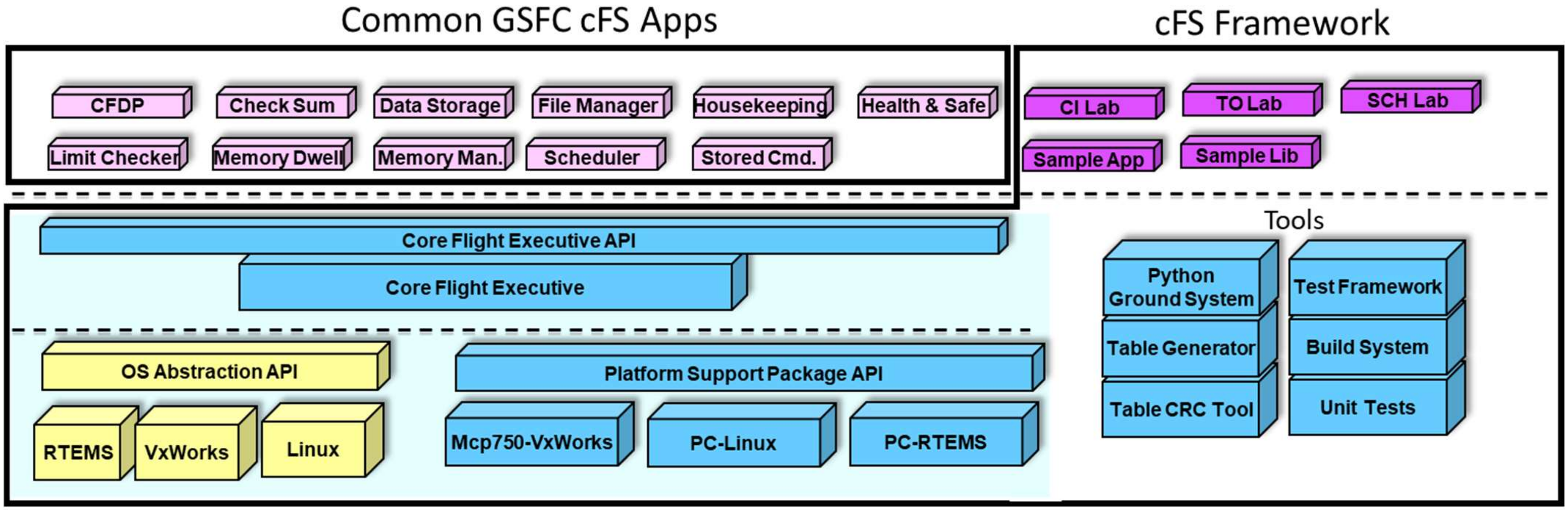
Real Time OS
Board Support Package
PROM Boot FSW

Tools

Python Ground System
Application Generator
Performance Tools
Performance Analyzer
Unit Tests
Build System

🟦 cFE Open Source Release 🟨 OSAL Open Source Release 🟪 Application Open Source Releases 🟦 3rd Party ⬜ Mission Developed

cFS Organization



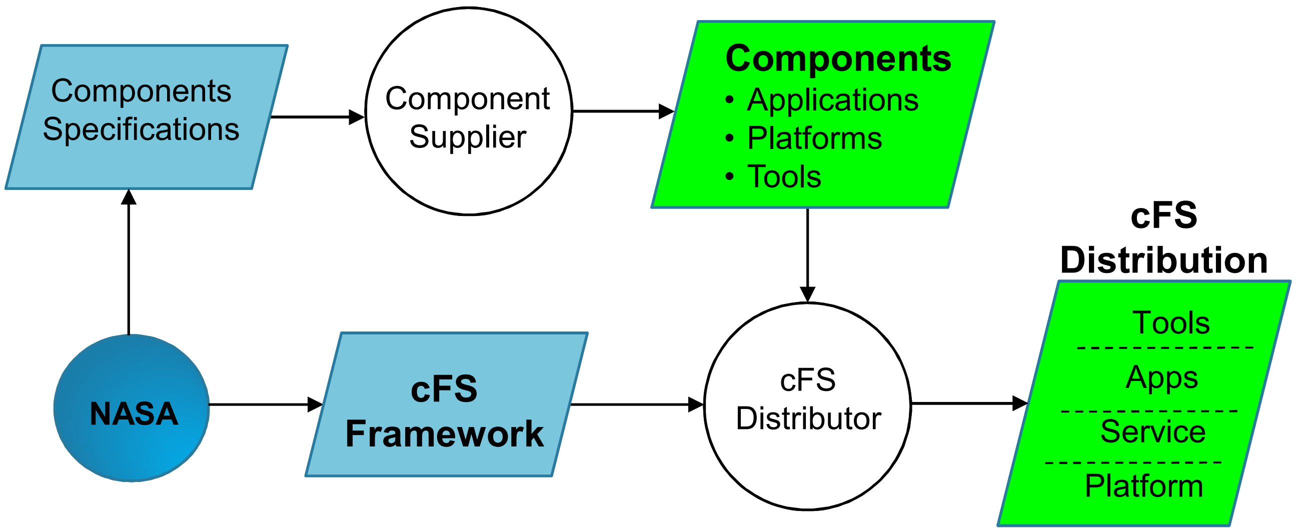
🟦 cFE Framework Open Source Release 🟨 OSAL Open Source Release ▢ GSFC Application Open Source Releases 🟪 Framework Apps

Key Definitions

- **Framework – The set of individual services, applications, tools, and infrastructure supported by the open source community CCB.**
- **Bundle – An executable version of the framework configured for a nominal Linux system. Links compatible versions of the framework elements as a recommended starting point for new cFS-based systems.**
- **Component – An individual application, service, or tool that can be used in a cFS-based system**
- **Distribution – A set of custom components packaged together with the framework; generally created and provided by a cFS user (individual or group) with specific needs (e.g. a NASA center, the GSFC SmallSat Project Office)**
- **cFE vs cFS:**
  - cFE is the Core Flight Executive services and API
  - cFS is a general collective term for the framework and the growing set of components

cFS Community

Community-based 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

Community-based Product Model

- **Community component supplier value proposition**
  - As the number of supported platforms increases then apps become more valuable
  - As the number of apps increases then supporting a cFS platform becomes more valuable
- **In 2019 vendors started to offer processor boards integrated with the cFS**
  - AI Tech partnering with Embedded Flight Systems to offer the cFS integrated on the SP0-S Single Board Computer
  - Genesis Engineering developing an integrated GEN6000 (SpaceCube 2.0) cFS product
  - Genesis pursuing a Space Act Agreement (SAA) that would include the creation of a platform certification test suite
- **Community members release, maintain, and distribute their apps**
  - Typically done via git
  - No one has established an "app store"

User Responsibilities

- **The cFS Framework has a NASA NPR-7150.2C Class E classification**

  "Software developed to explore a design concept or hypothesis but not used to make decisions for an operational Class A, B, or C system or to-be-built Class A, B, or C system"

  - The cFS Framework provides artifacts to support Class B missions and a subset of artifacts to support Class A missions
  - End-users are responsible for classifying the software system that uses the cFS Framework
- **End-users are responsible for complying with International Traffic in arms Regulations (ITAR)**
- **Projects are responsible for verifying all of their requirements**
  - Many projects treat cFS in the same way as operating systems

Obtaining cFS “Products”

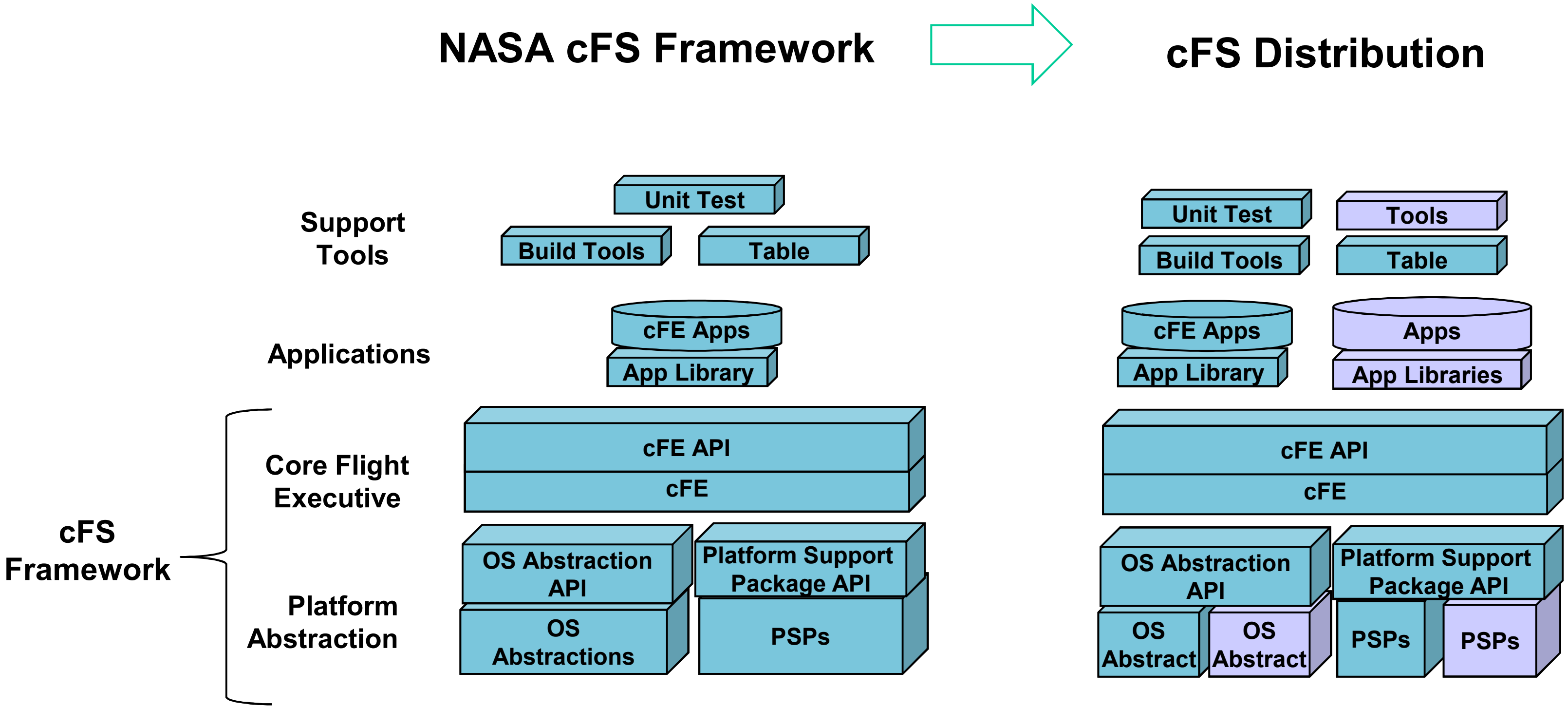
- **cFS Bundle**
  - Contains the cFS Framework packaged with additional components to create a system that can easily be built, executed, and unit tested on a Linux platform
  - http://github.com/nasa/cFS

- **User Components**
  - Search https://github.com/nasa/ or do a general web search on NASA cFS

- **Distributions**
  - Listed on a later slide
  - Some distributions contain many of the common apps which give you a good starting point for apps

- **Engage with the Community**
  - Ask the community mailing list (See backup slides)
    - Especially useful when porting to a new platform
  - Contact a cFS team member (See backup slides)

cFS Product Model



- The NASA Configuration Control Board (CCB) manages the “cFS Framework”
- “cFS Distribution” created by augmenting the NASA cFS Framework with components (platforms, apps, and tools) to create an operational system

cFS Distributions

Name/Link	Intended Audience	Overview
cFS Framework-101	cFS Framework training package	This is a training tool for individuals to learn how to develop software with NASA-developed Core Flight software (CFS) framework. No agreement is necessary through this catalog. Training is created by JSC and is open source.
cFS Bundle	Initial cFS build for a developer or a project	This repository contains submodules for the cFE, OSAL, and apps, as well as instructions for building the system. This distribution has been compiled/linked but has not been verified as an operational system.
NASA Operational Simulator for Small Satellites (NOS3)	Initial cFS platform for a project	NOS3 provides a complete cFS system designed to support satellite flight software development throughout the project life cycle. It includes • 42 Spacecraft dynamics and visualization, NASA GSFC • cFS – core Flight System, NASA GSFC • COSMOS – Ball Aerospace • ITC Common – Loggers and developer tools, NASA IV&V ITC • NOS Engine – Middleware bus simulator, NASA IV&V ITC
OpenSatKit (OSK)	cFS training platform for new cFS developers	OSK provides a complete cFS system to simplify the cFS learning curve, cFS deployment, and application development. The kit combines three open source tools to achieve these goals: • cFS – core Flight System, NASA GSFC • COSMOS – command and control platform for embedded systems, Ball Aerospace • 42 dynamic simulator, NASA GSFC

Community Operational Procedures

- **Version Control**
  - Master Branch
  - Integration Candidates
  - Release Candidates

- **User Contributions**
  - Community Contribution process and Contributor License Agreement (CLA)

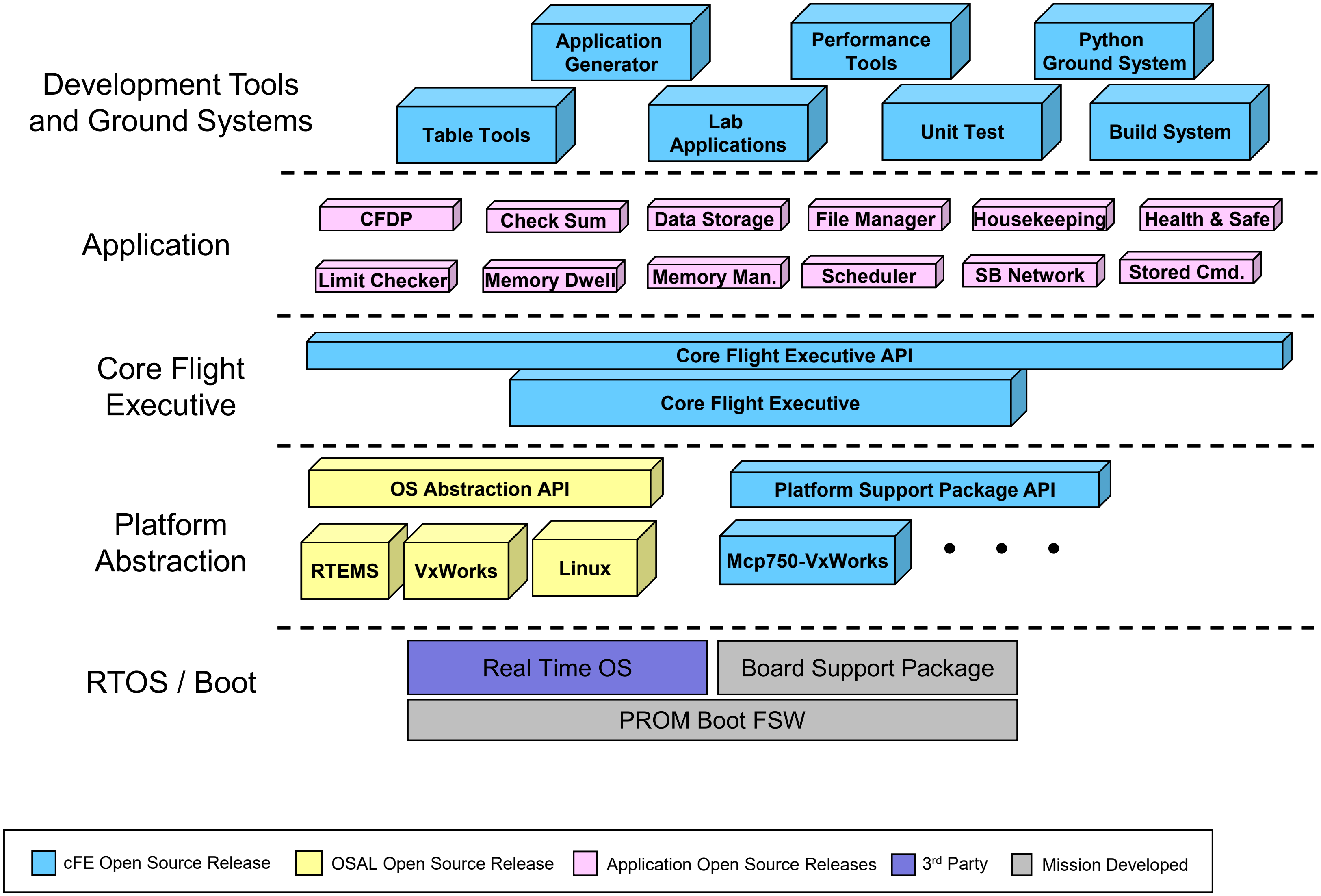
- **Feature Deprecation**
  - Mark feature as deprecated on any release
  - Provide tools/process that will warn applications when a feature is marked as deprecated
  - Only deprecate on major versions

Core Flight System Architectural Overview

Architecture Goals

1. Reduce time to deploy high quality flight software
2. Reduce project schedule and cost uncertainty
3. Directly facilitate formalized software reuse
4. Enable collaboration across organizations
5. Simplify sustaining engineering (AKA. On Orbit FSW maintenance) Missions last 10 years or more
6. Scale from small instruments to Hubble class missions
7. Build a platform for advanced concepts and prototyping
8. Create common standards and tools across the center

cFS Architecture Layers



Development Tools and Ground Systems

Application Generator | Performance Tools | Python Ground System

Table Tools | Lab Applications | Unit Test | Build System

Application

CFDP | Check Sum | Data Storage | File Manager | Housekeeping | Health & Safe

Limit Checker | Memory Dwell | Memory Man. | Scheduler | SB Network | Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API | Platform Support Package API

RTEMS | VxWorks | Linux | Mcp750-VxWorks • • •

RTOS / Boot

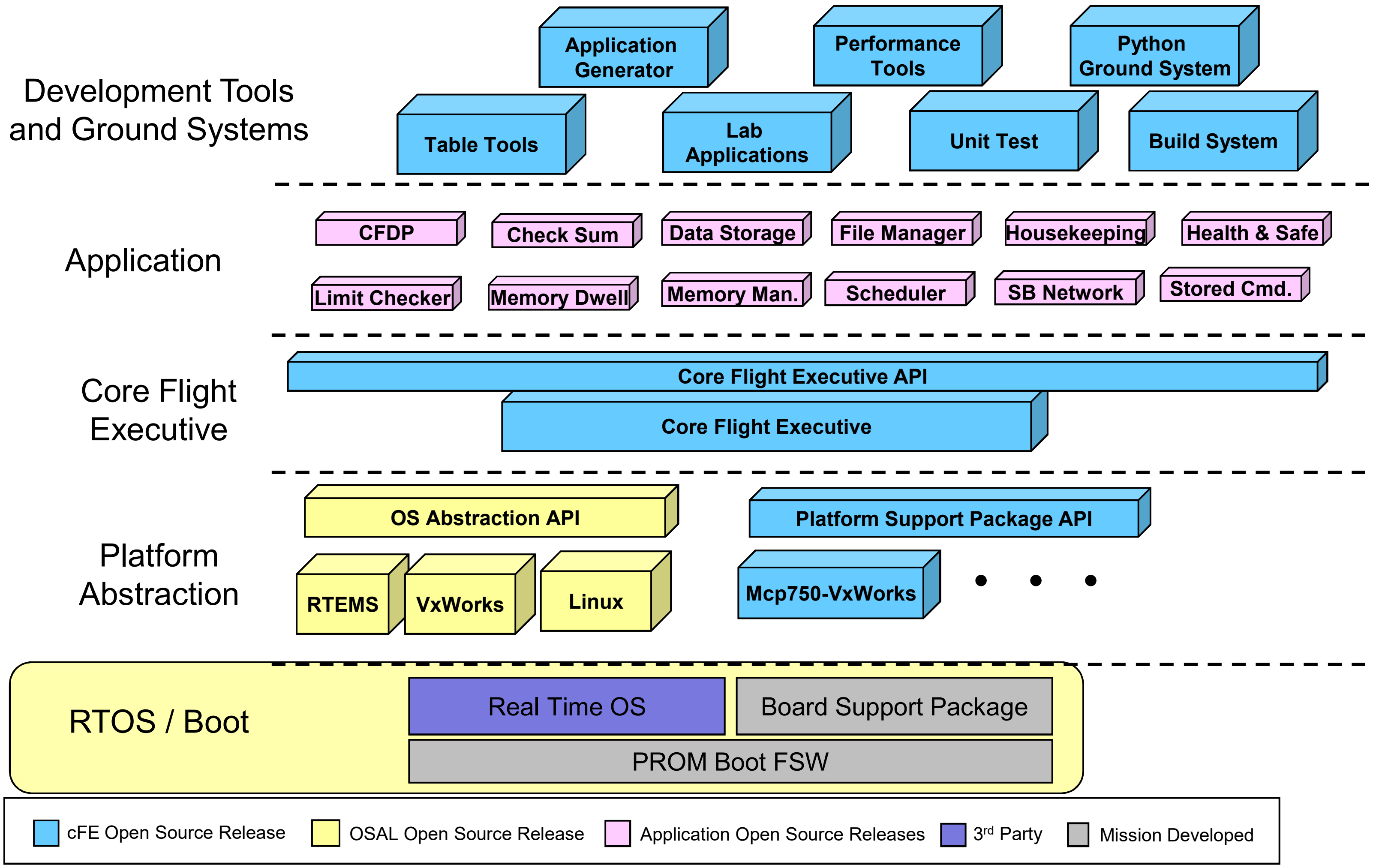
Real Time OS | Board Support Package

PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release 🟥 Application Open Source Releases 🟪 3rd Party ⬜ Mission Developed

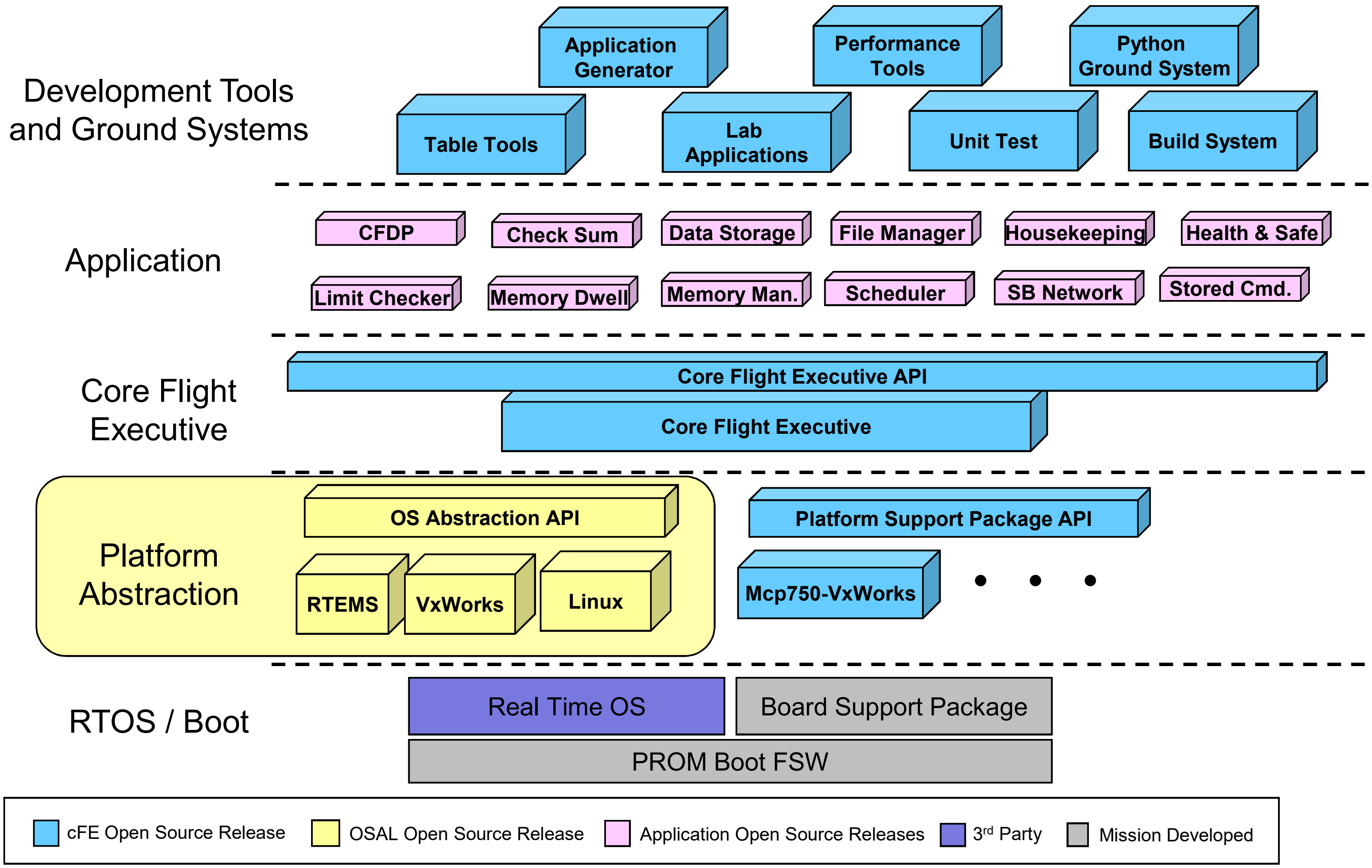
Operating System / Boot Layer

Provides the commercial, open-source, or custom software interface between the processor and the FSW. Real-time multi-tasking preemptive scheduling operating systems used for flight applications.



Platform Abstraction - OSAL

The OS Abstraction Layer (OSAL) is a software library that provides a single Application Program Interface (API) to the core Flight Executive (cFE) regardless of the underlying real-time operating system.



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

RTEMS

VxWorks

Linux

Platform Support Package API

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

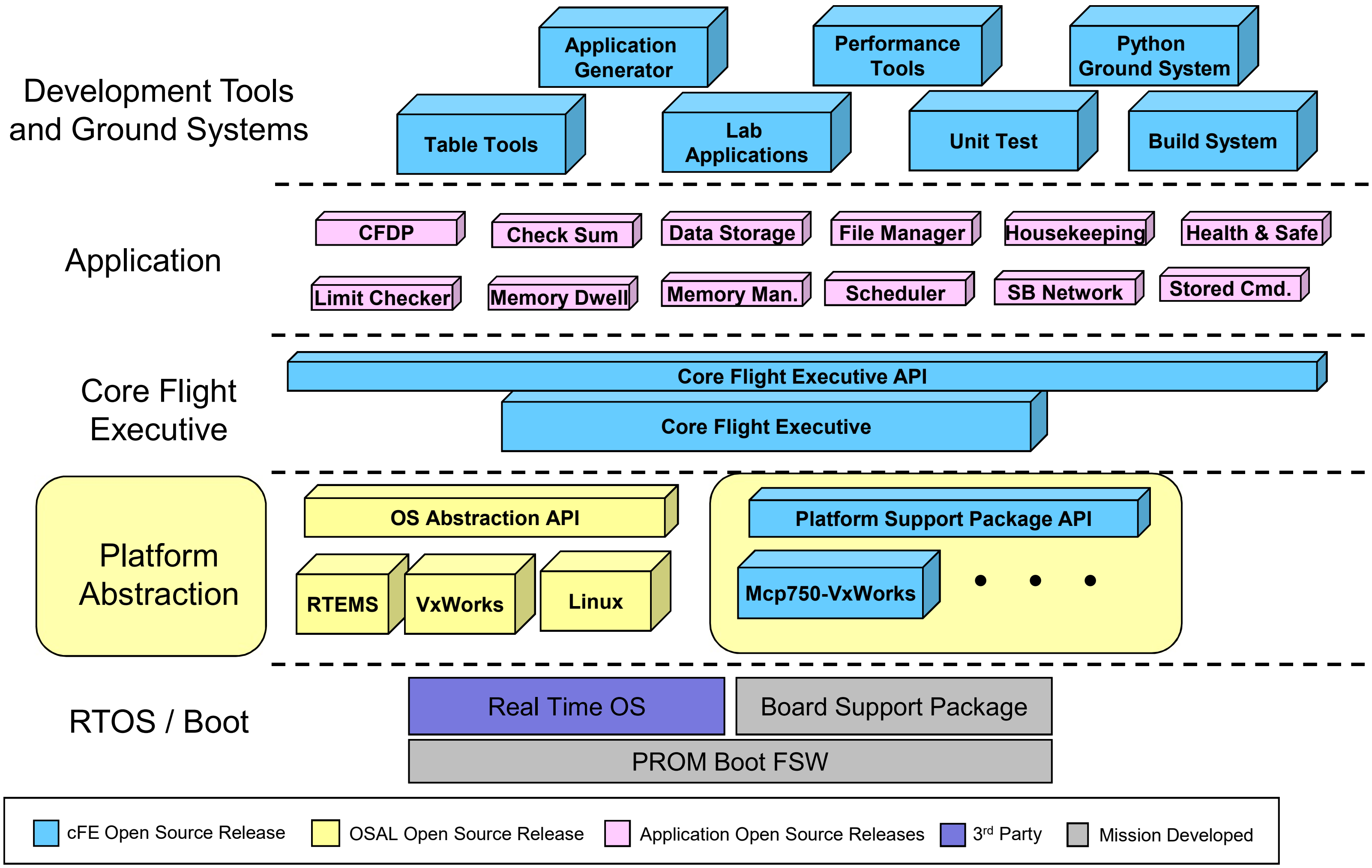
Board Support Package

PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release 🟪 Application Open Source Releases 🟦 3rd Party ⬜ Mission Developed

Platform Abstraction - PSP

The Platform Support Package (PSP) is a software library that provides a single Application Program Interface (API) to underlying avionics hardware and board support package.



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

RTEMS

VxWorks

Linux

Platform Support Package API

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

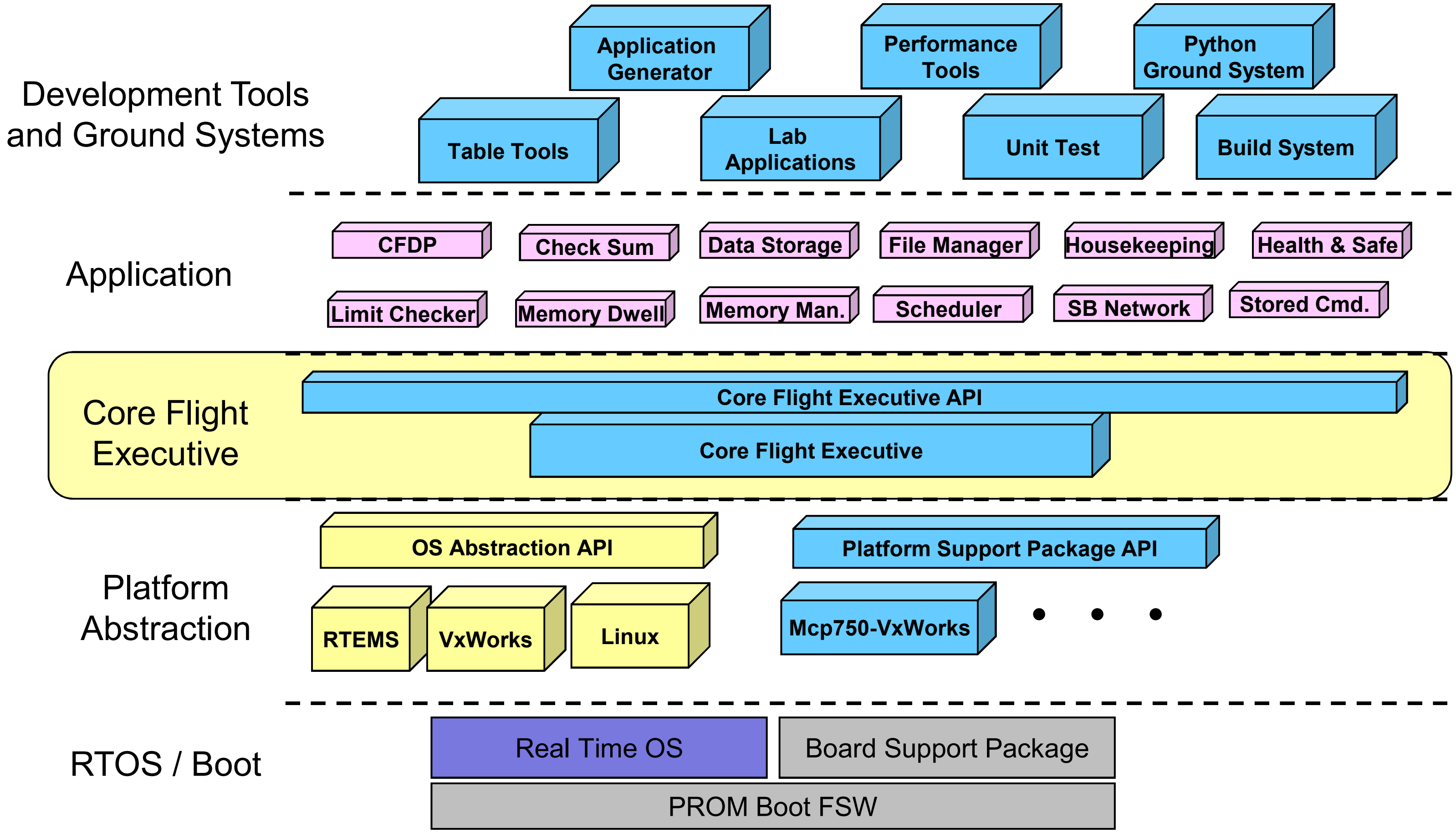
Board Support Package

PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release ▨ Application Open Source Releases 🟪 3rd Party ⬜ Mission Developed

Core Flight Executive

The cFE is a portable, platform-independent framework that creates an application runtime environment by providing services that are common to most flight applications.



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

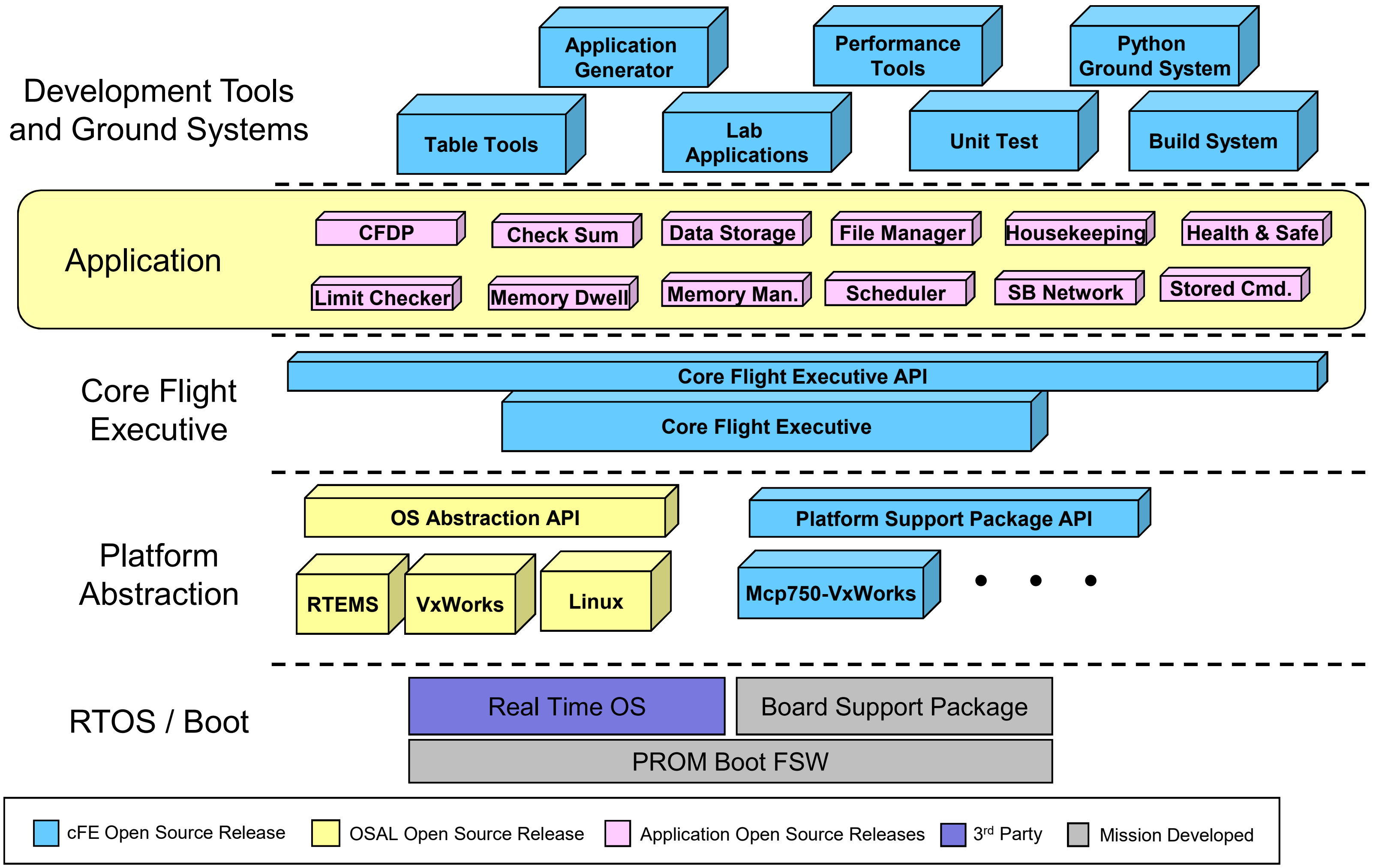
Board Support Package

PROM Boot FSW

■ cFE Open Source Release ■ OSAL Open Source Release ■ Application Open Source Releases ■ 3rd Party ■ Mission Developed

Applications

Applications provide mission functionality using a combination of cFS community apps and mission-specific apps.



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

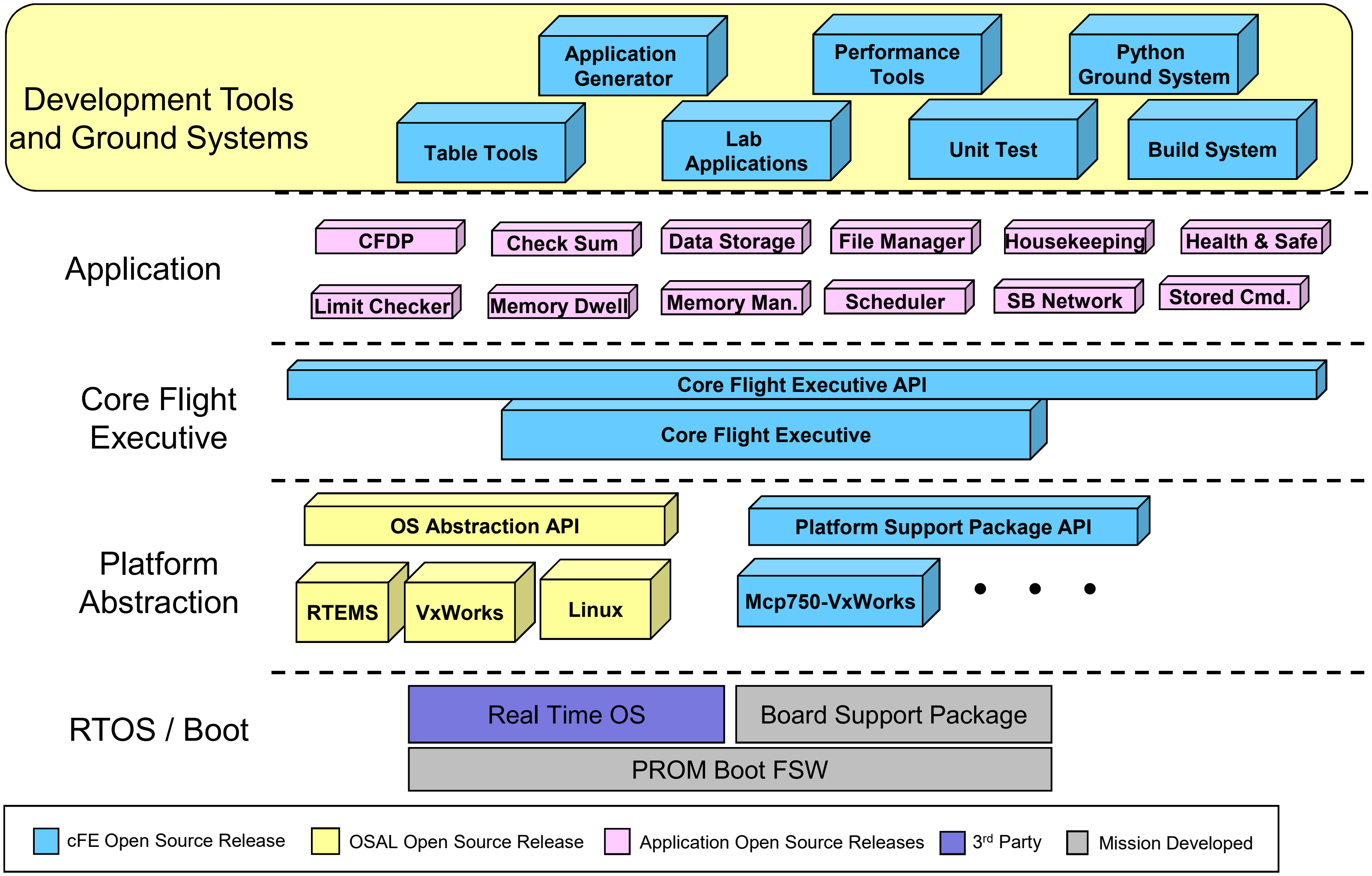
Board Support Package

PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release ▢ Application Open Source Releases 🟪 3rd Party ⬜ Mission Developed

Development Tools & Ground Systems

Development tools and ground systems are used to test and run the cFS. A variety of ground systems can be used with cFS. Ground system and tool selection generally vary by project.



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

Board Support Package

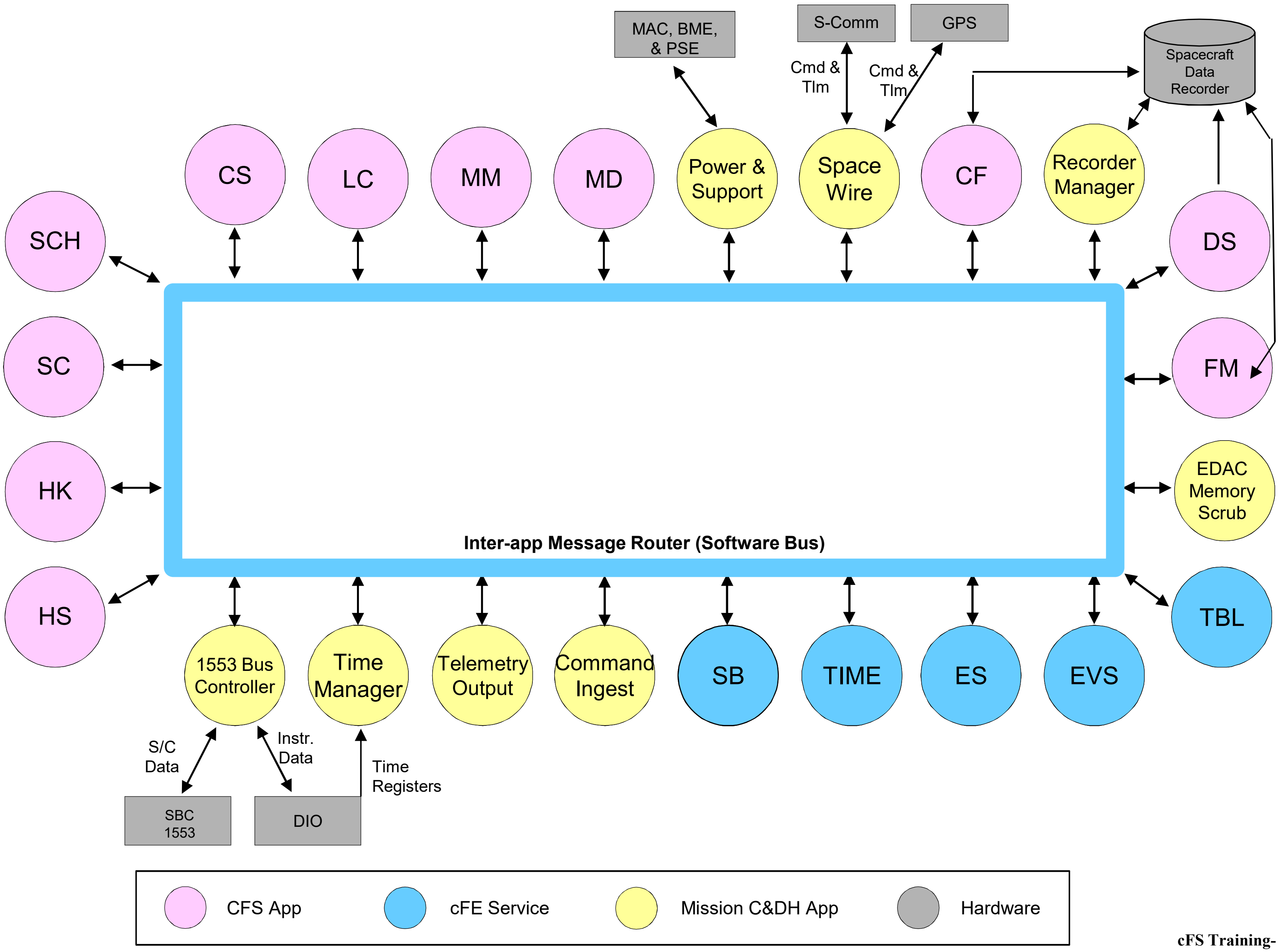
PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release ▨ Application Open Source Releases 🟪 3rd Party ⬜ Mission Developed

cFS Applications

- **Can run anywhere the cFS framework has been deployed**
- **GSFC has released 12 applications that provide common command and data handling functionality such as**
  - Stored command management and execution
  - Onboard data storage file management
- **Missions use a combination of custom and reused applications**

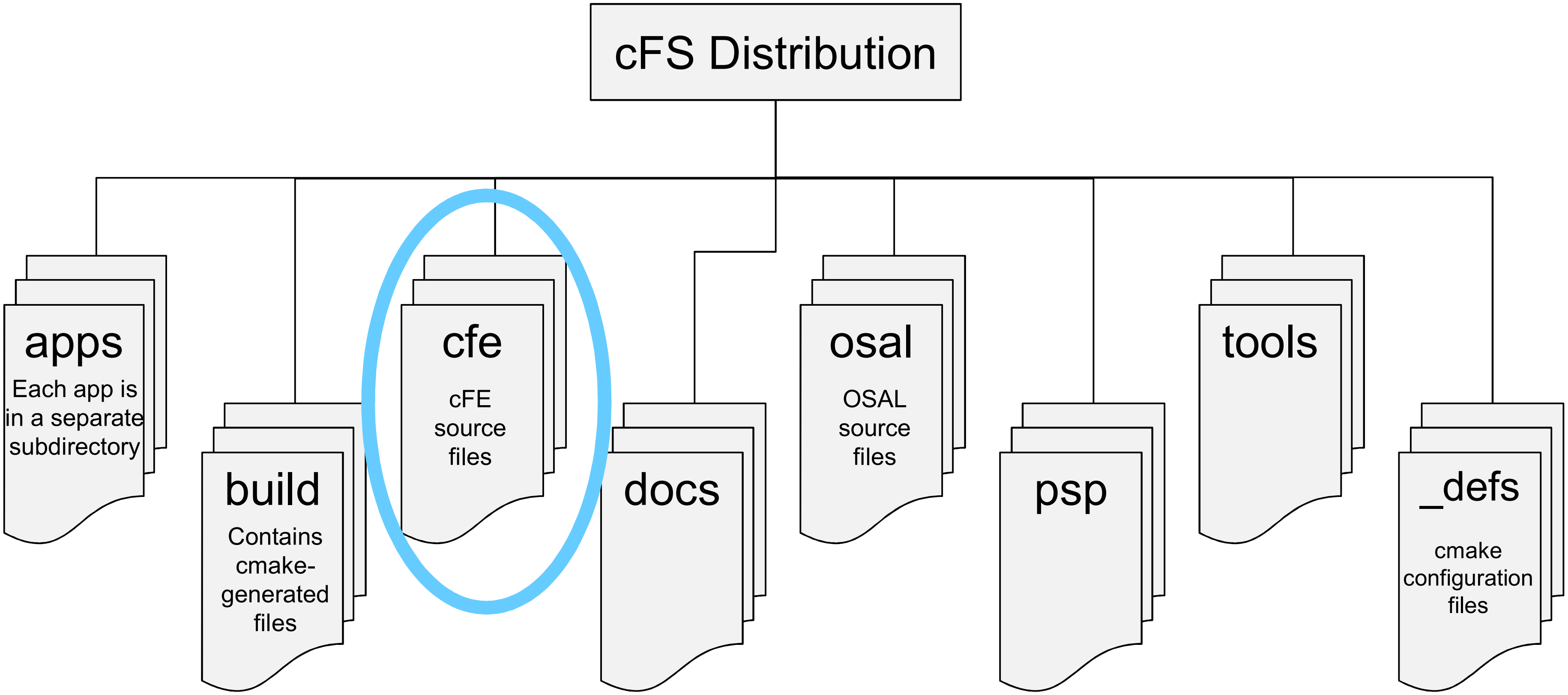
Mission Application Example



cFS Libraries

- **What is a library?**
  - A collection of utilities available for use by apps
  - No main task execution in the library
  - Exist at the application layer of the cFS
- **Specified in the cfe_es_startup.scr script and loaded at cFE startup**
- **Libraries can't use application services that require registration**
  - e.g. Event Services
- **Checksum can't do library code space. No cFE API.**

cFS Mission Directory Structure



cFS Distribution

apps
Each app is in a separate subdirectory

build
Contains cmake-generated files

cfe
cFE source files

docs

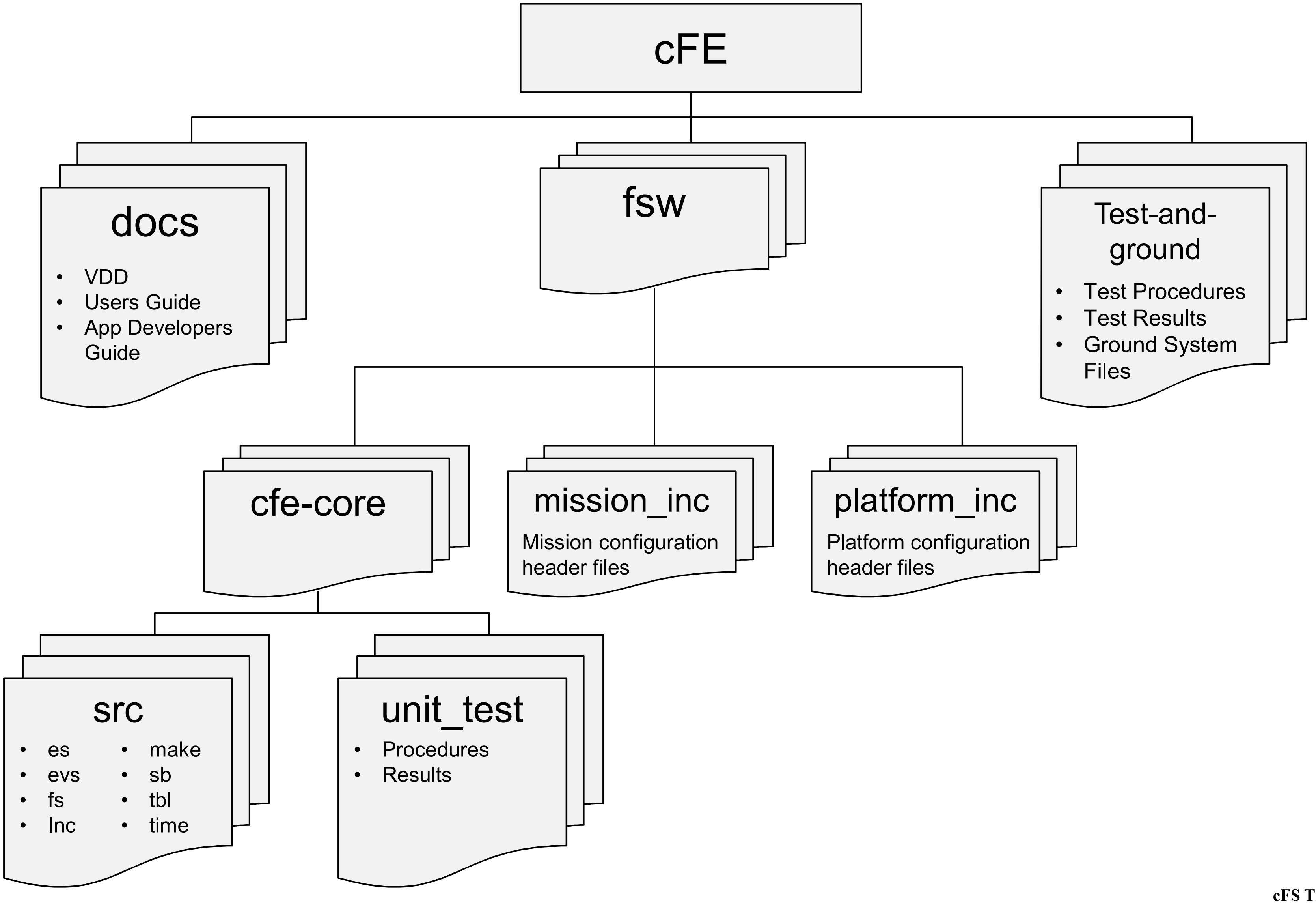
osal
OSAL source files

psp

tools

_defs
cmake configuration files

cFE Directory Structure



National Aeronautics and Space Administration

Module 1: Backup Charts

cFS References

Where is the cFS?

- **cFS Framework, http://github.com/nasa/cFS**
  - Source code
  - Requirements and user guides

- **OSAL, http://sourceforge.net/projects/osal/**
  - Source code
  - Requirements and user guides
  - Tools

- **Links to GSFC applications, https://cfs.gsfc.nasa.gov**

GSFC Open Source Apps

Application	Function
CFDP	Transfers/receives file data to/from the ground
Checksum	Performs data integrity checking of memory, tables and files
Command Ingest Lab	Accepts CCSDS telecommand packets over a UDP/IP port
Data Storage	Records housekeeping, engineering and science data onboard for downlink
File Manager	Interfaces to the ground for managing files
Housekeeping	Collects and re-packages telemetry from other applications.
Health and Safety	Ensures critical tasks check-in, services watchdog, detects CPU hogging, calculates CPU utilization
Limit Checker	Provides the capability to monitor values and take action when exceed threshold
Memory Dwell	Allows ground to telemeter the contents of memory locations. Useful for debugging
Memory Manager	Provides the ability to load and dump memory
Software Bus Network	Passes Software Bus messages over various “plug-in” network protocols
Scheduler	Schedules onboard activities via (e.g. HK requests)
Scheduler Lab	Simple activity scheduler with a one second resolution
Stored Command	Onboard Commands Sequencer (absolute and relative)
Stored Command Absolute	Allows concurrent processing of up to 5 (configurable) absolute time sequences
Telemetry Output Lab	Sends CCSDS telemetry packets over a UDP/IP port

References

- **Open Source**
  - OSAL 4.2.0: http://sourceforge.net/projects/osal/
  - cFE 6.5.0: http://sourceforge.net/projects/coreflightexec
  - Goddard: http://opensource.gsfc.nasa.gov
  - NASA: http://code.nasa.gov

- **Goddard's Strategic Partnership Office**
  - https://partnerships.gsfc.nasa.gov/index.html

- **cFS Websites and Publications**
  - https://cfs.gsfc.nasa.gov
  - Publications
    - Software Architecture Review Board (SARB) Review and Assessment of Goddard Space Flight Center's (GSFC's) core Flight Executive/Core Flight System (cFE/cFS), https://nen.nasa.gov/web/software/sarb
    - Verifying Architectural Design Rules of the Flight Software Product Line, Dharmalingam Ganesan, Mikael Lindvall, Chris Ackermann *Fraunhofer CESE,* http://www.fc-md.umd.edu/save
    - LINUX JOURNAL
    - Ask Magazine
    - AETD Monthly Message

Module 1: Backup Charts

Architecture

Quality Analysis - 1

- **Operability**
  - The architecture must enable the flight system to operate in an efficient and understandable way
- **Reliability**
  - The architecture implementation must be known to behave correctly in nominal and expected off-nominal situations
- **Robustness**
  - The architecture implementation must be predictable and safe in the presence of unexpected conditions
- **Performance**
  - The architecture implementation must be efficient in runtime resources given the targeted processing environments
- **Testability**
  - The architecture implementation must be easily and comprehensively testable in situ in flight like scenarios
- **Maintainability**
  - The architecture implementation must be maintainable in the operational environment

Quality Analysis - 2

- **Effective Reuse**
  - The architecture must support an effective reuse approach. This includes the software and artifacts (e.g. requirements, design, code, review presentations, tests, operations guides, command and telemetry databases). The goal is to achieve 100% reuse of a software component with no code changes.
- **Composability**
  - Properties established at the component level, such as interfaces, timeliness or testability, also hold at the system level. For an application or node to be composable the architecture and process must support:
    - Independent development of nodes
    - Integration of the node into a system should not invalidate services in the value and temporal domains
    - Integration of an additional node into a functioning system should not disturb the correct operation of the existing nodes
    - Replica determinism – identical copies of nodes must produce identical results in an identical order, within a specified time interval
- **Predicable Development Schedule**
  - Development estimates provided by the FSW team should be reliable

Quality Analysis - 3

- **Scalability**
  - The FSW must scale with mission requirements. (Example: instruments or subsystem processor may only need a small amount of message buffer space. This should be configurable to avoid wasting memory resources.)
- **Adaptability**
  - The FSW must be capable of supporting a range of platforms and missions.
- **Minimized Development Cost**
  - Costs for mission functions should be as low as possible. The teams must consider the difference between NRE and costs for a given mission.
- **Technology infusion**
  - The FSW should support the infusion of new hardware and software technologies with minimal side effects.

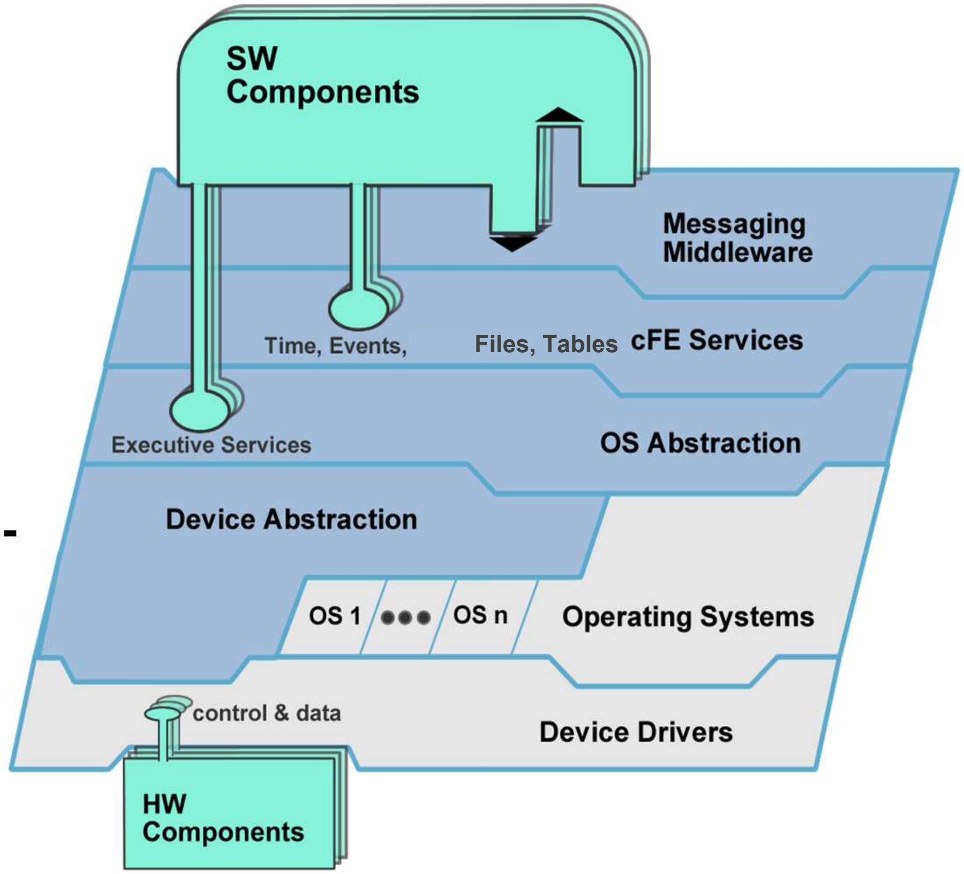
Layered Service Architecture

- Each layer and service has a standard API.

- Each layer “hides” its implementation and technology details.

- Internals of a layer can be changed -- without affecting other layers’ internals and components.

- Provides Middleware, OS and HW platform-independence.



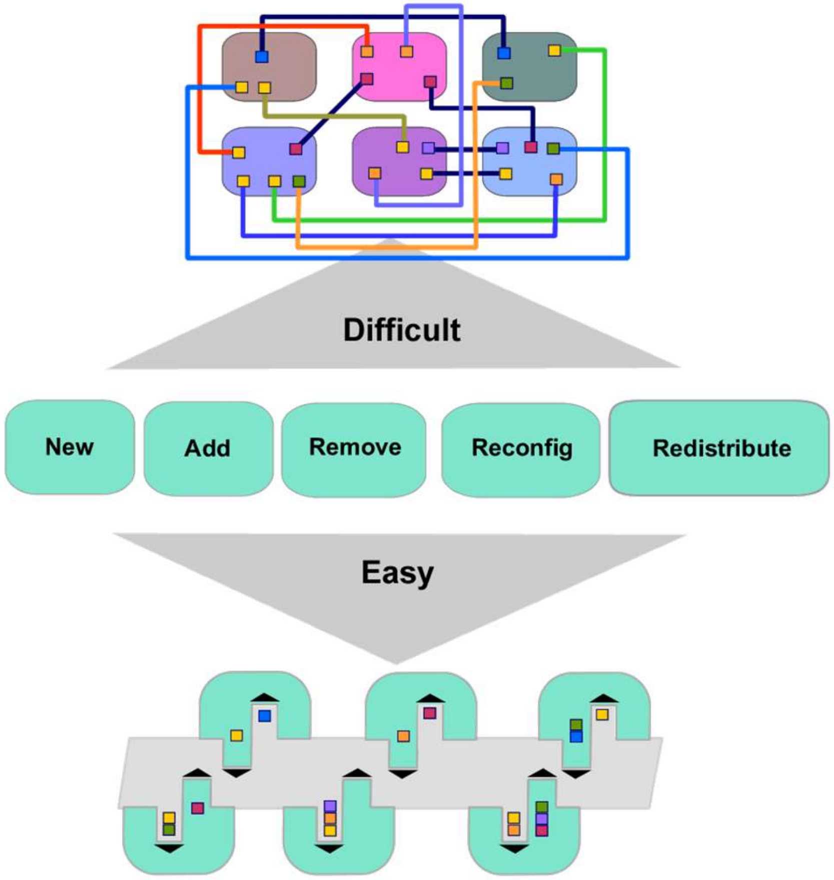
Plug and Play

Plug and Play

- cFE APIs support add and remove functions.
- SW components can be switched in and out at runtime, without rebooting or rebuilding the system SW.
- Qualified Hardware and cFS-compatible software both "plug and play".

Impact

- Changes can be made dynamically during development, test and on-orbit even as part of contingency management.
- Technology evolution/change can be taken advantage of later in the development cycle.
- Testing environment is flexible (can use different GSE, test apps, simulators, etc.).



This powerful paradigm allows SW components to be switched in and out at runtime, without rebooting or rebuilding the system SW.

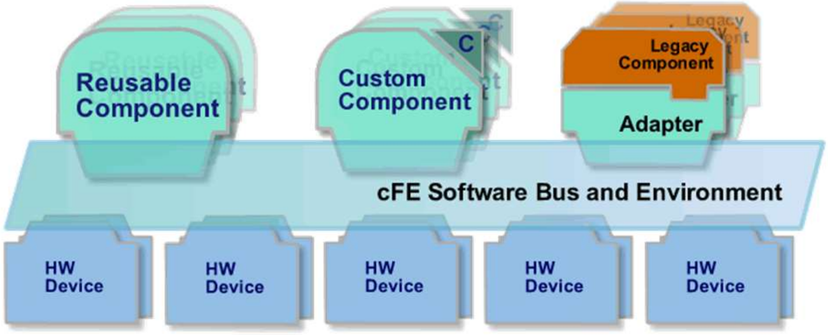
Reusable Components

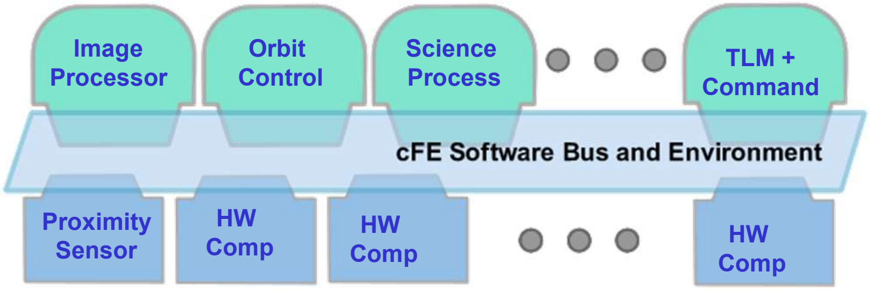
Reusable Components

- Common FSW functionality has been abstracted into a library of reusable components and services.
- Components are tested and documented.
- A system is built from:
  - Core services
  - Reusable components
  - Custom mission specific components
  - Adapted legacy components

Impact:

- Reuse of tested, certified components supplies savings in each phase of the software development cycle.
- Reduces risk.
- Teams focus on the custom aspects of their project and don’t “reinvent the wheel”.





Component Example

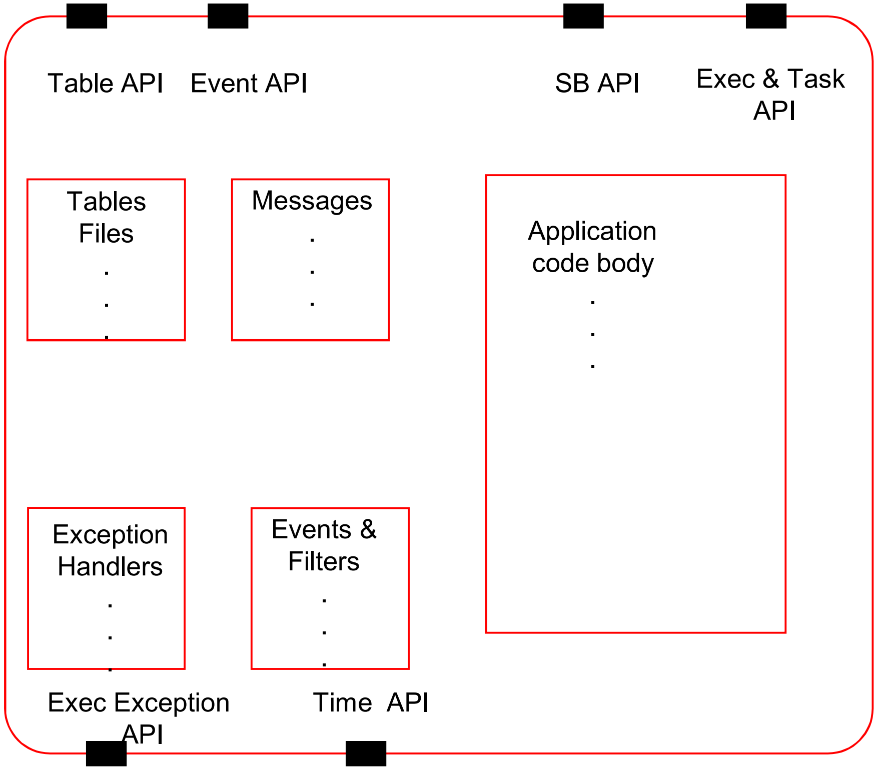
- Interface only through core APIs.

- A component contains all data needed to define its operation.

- Components register for services
  - Register exception handlers
  - Register Event counters and filter
  - Register Tables
  - Publish messages
  - Subscribe to messages

- Component may be added or removed at runtime. (Allows rapid prototyping during development)

- Configuration parameters allow tailoring of components



National Aeronautics and Space Administration

Core Flight System (cFS) Training

Module 2: Core Flight Executive (cFE)

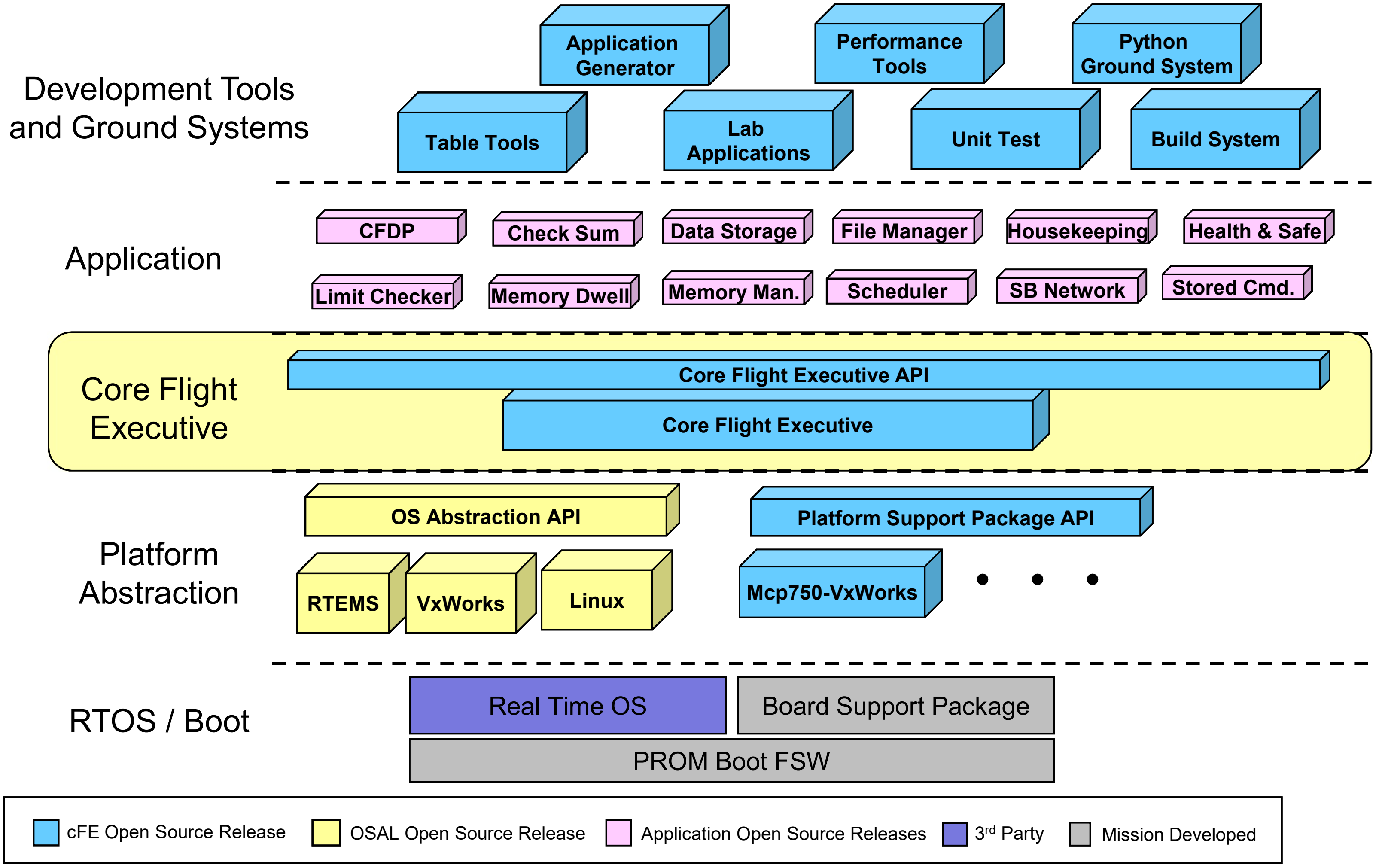
Services

August 3, 2019

Course Agenda

1. **Introduction**
2. **cFE Services**
   a) Executive Services
   b) Time Services
   c) Event Services
   d) Software Bus
   e) Table Services
3. **Application Layer**
   a) cFS Applications
   b) cFS Libraries

cFE Services - cFS Context



What are the cFE Services?

Executive Services (ES)

- Manage the software system and create an application runtime environment

Time Services (TIME)

- Manage spacecraft time

Event Services (EVS)

- Provide a service for sending, filtering, and logging event messages

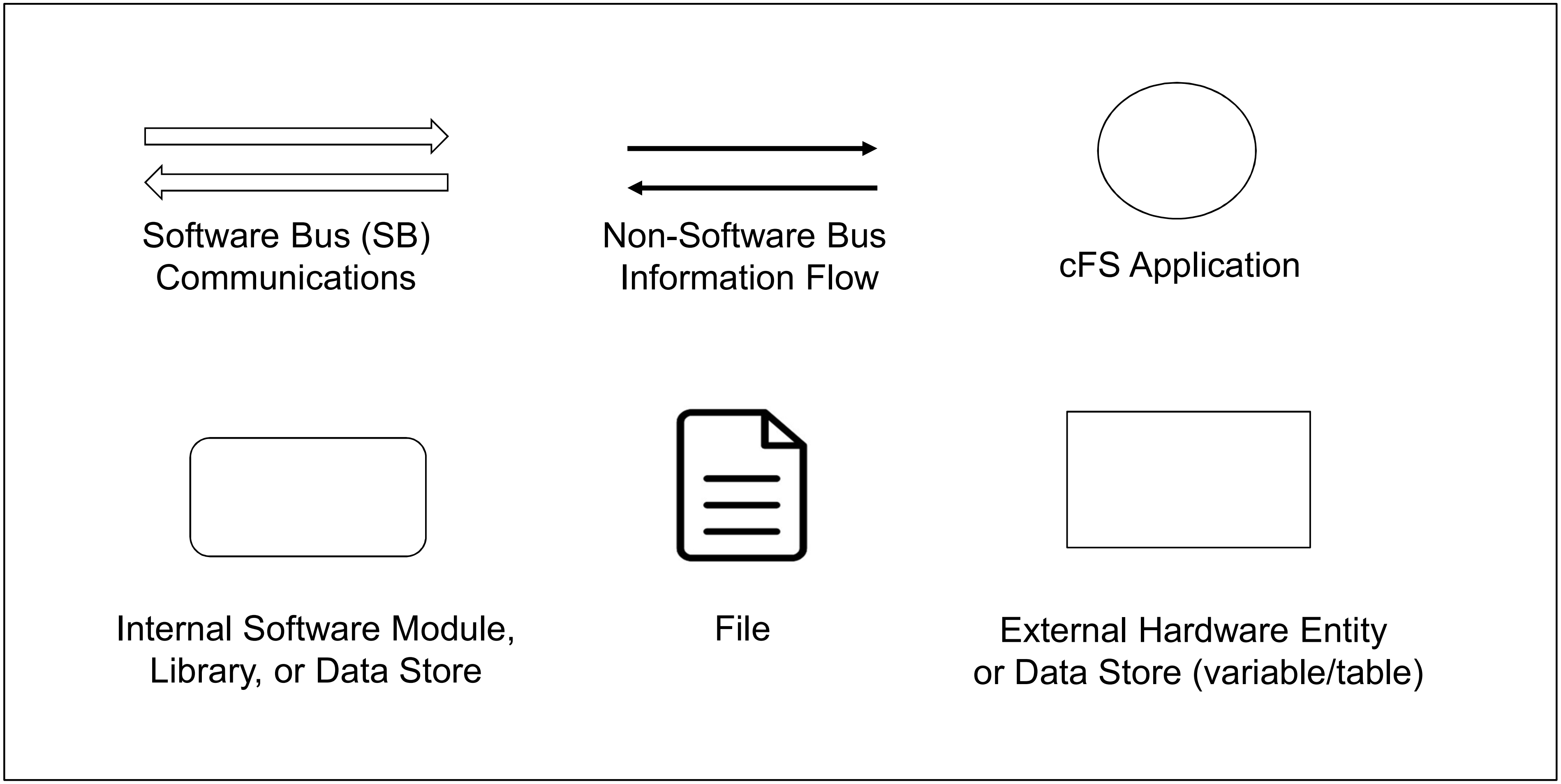
Software Bus (SB) Services

- Provide an application publish/subscribe messaging service

Table Services (TBL)

- Manage application table images

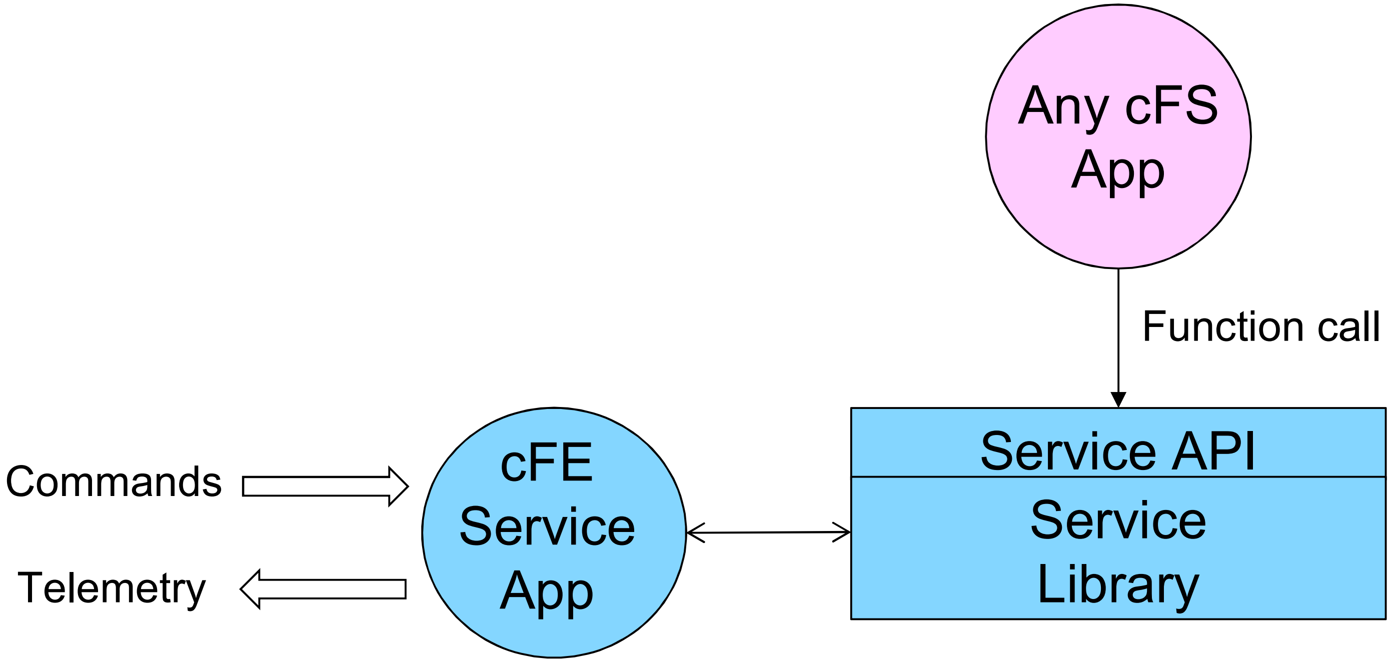
Diagram Notation

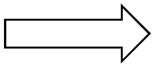

Software Bus (SB) Communications
Non-Software Bus Information Flow
cFS Application
Internal Software Module, Library, or Data Store
File
External Hardware Entity or Data Store (variable/table)

- Common data flows such as command inputs to an app and telemetry outputs from an app are often omitted from context diagrams unless they are important to the particular situation

Common cFE Service Design

- **Each cFE service has:**
  - A library that is used by applications
  - An application that provides a ground interface for operators to use to manage the service





Application Runtime Environment

- **cFE Services provide an Application Runtime Environment**
- **The cFE service API provides a functional interface to use the services**
  - Very stable. No functional change since 2008
- **Obtaining information beyond the housekeeping packet**
  - Commands to send one time telemetry packets
  - Commands to write onboard service configuration data to files

Application-Centric Architecture

- **Applications are an architectural component that owns cFE and operating system resources**
- **Resources are acquired during initialization and released when an application terminates**
  - Helps achieve the architectural goal for a loosely coupled system that is scalable, interoperable, testable (each app is unit tested), and maintainable
- **Concurrent execution model**
  - Each app has its own execution thread and apps can spawn child tasks
- **The cFE service and Platform Abstraction APIs provide a portable functional interface**
- **Write once run anywhere the cFS framework has been deployed**
  - Defer embedded software complexities due to cross compilation and target operating systems
  - Framework provides seamless application transition from technology efforts to flight projects
- **Reload apps during operations without rebooting**

Configuration Parameter Scope

- **Mission configuration parameters – used for ALL processors in a mission (e.g. time epoch, maximum message size, etc.)**
  - Default contained in:
    - \cfe\fsw\mission_inc\cfe_mission_cfg.h
    - \apps\xx\fsw\mission_inc\xx_mission_cfg.h, xx_perfids.h

- **Platform Configuration parameters – used for the specific processor (e.g. time client/server config, max number of applications, max number of tables, etc.)**
  - Defaults contained in:
    - \cfe\fsw\platform_inc\cpuX\cfe_platform_cfg.h, cfe_msgids_cfg.h
    - \apps\xx\fsw\platform_inc\xx_platform_cfg.h, xx_msgids.h
    - \osal\build\inc\osconfig.h

- **Just because something is configurable doesn’t mean you want to change it**
  - E.g. CFE_EVS_MAX_MESSAGE_LENGTH

Unique Identifier Configuration Parameters

- **Software Bus Message Identifiers**
  - cfe_msgids.h (message IDs for the cFE should not have to change)
  - app_msgids.h (message IDs for the Applications) are platform configurations

- **Executive Service Performance Identifiers**
  - cFE performance IDs are embedded in the core
  - app_perfids.h (performance IDs for the applications) are mission configuration

- **Task priorities are not configuration parameters but must be managed from a processor perspective**

- **Note cFE strings are case sensitive**

cFS Application Mission and Platform Configuration Files

File	Purpose	Scope	Notes
cfe_mission_cfg.h	cFE core mission wide configuration	Mission	
cfe_platform_cfg.h	cFE core platform configuration	Platform	Most cFE parameters are here
cfe_msgids.h	cFE core platform message IDs	Platform	Defines the message IDs the cFE core will use on that Platform(CPU)
osconfig.h	OSAL platform configuration	Platform	
XX_mission_cfg.h	A cFS Application's mission wide configuration	Mission	Allows a single cFS application to be used on multiple CPUs on one mission
XX_platform_cfg.h	Application platform wide configuration	Platform	
XX_msgids.h	Application message IDs	Platform	
XX_perfids.h	Application performance IDs	Platform	

Exercise 1 – Build and Run the cFE

Part 1 - Setup

To setup the cFS Bundle directly from the latest set of interoperable repositories:

```
    git clone https://github.com/nasa/cFS.git
    cd cFS
    git submodule init
    git submodule update
```

Copy in the default makefile and definitions:

```
    cp cfe/cmake/Makefile.sample Makefile
    cp -r cfe/cmake/sample_defs sample_defs
```

If running on a standard linux build as a normal user, define OSAL_DEBUG_PERMISSIVE_MODE for best effort message queue depth and task priorities.

```
    sed -i 's/undef OSAL_DEBUG_PERMISSIVE_MODE/define OSAL_DEBUG_PERMISSIVE_MODE/g'
sample_defs/default_osconfig.h
```

Exercise 1 – Build and Run the cFE

Part 2 – Build and Run

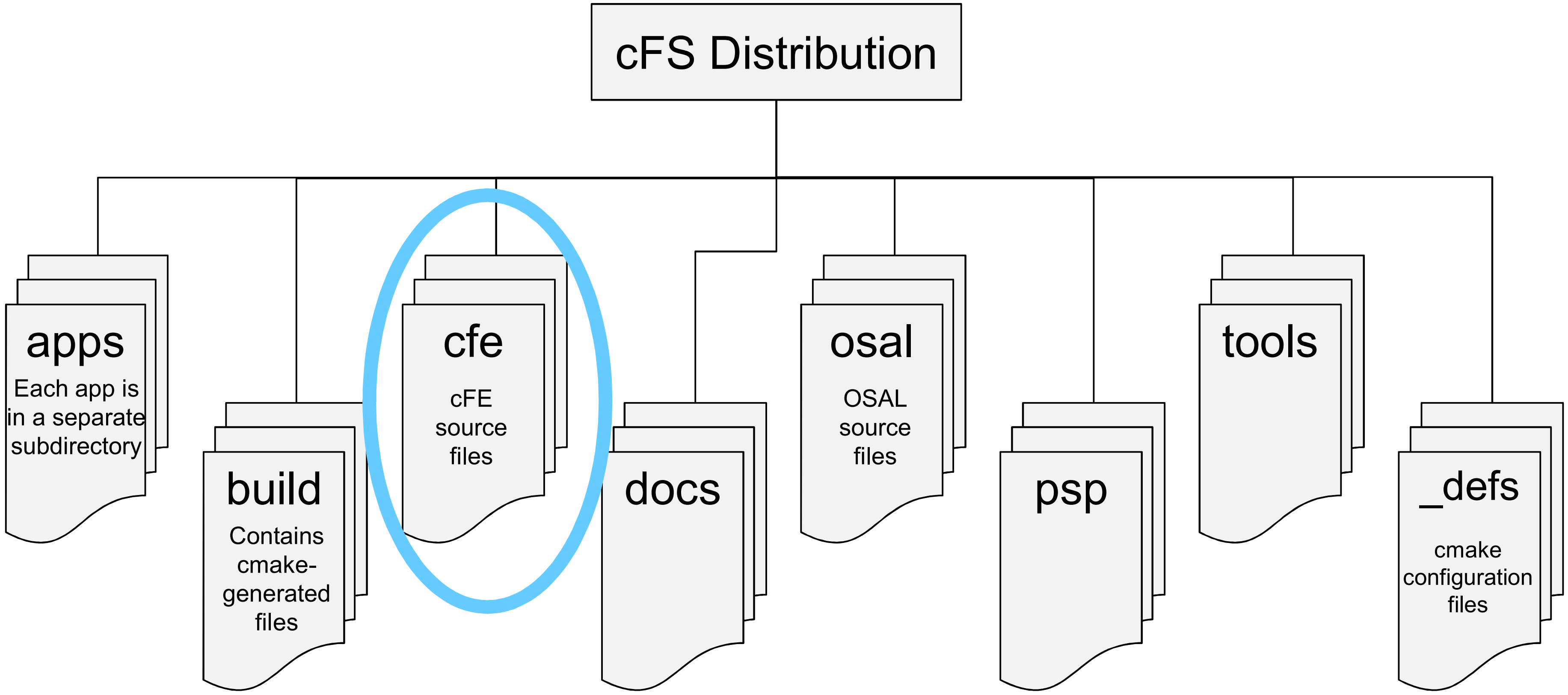
The cFS Framework including sample applications will build and run on the pc-linux platform support package (should run on most Linux distributions), via the steps described in https://github.com/nasa/cFE/tree/master/cmake/README.md. Quick-start is below:

To prep, compile, and run (from cFS directory above):

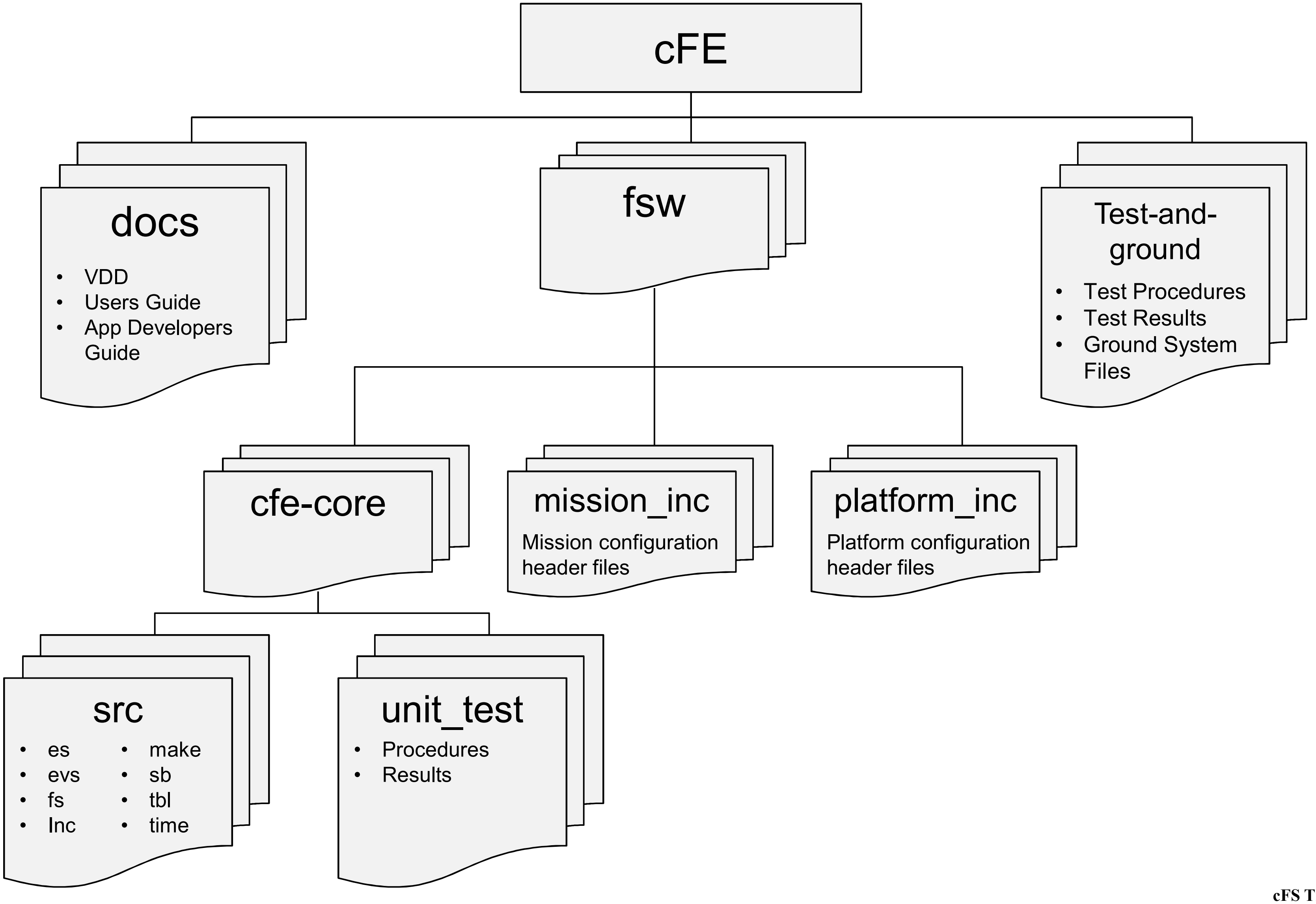
```
make prep
make
make install
cd build/exe/cpu1/
./core-cpu1
```

Should see startup messages, and CFE_ES_Main entering OPERATIONAL state. Note the code must be executed from the build/exe/cpu1 directory to find the startup script and shared objects.

Exercise 1 Recap - cFS Mission Directory Structure


cFS Distribution
apps
Each app is in a separate subdirectory
build
Contains cmake-generated files
cfe
cFE source files
docs
osal
OSAL source files
psp
tools
_defs
cmake configuration files

Exercise 1 Recap - cFE Directory Structure



cFE

docs

- VDD
- Users Guide
- App Developers Guide

fsw

Test-and-ground

- Test Procedures
- Test Results
- Ground System Files

cfe-core

mission_inc

Mission configuration header files

platform_inc

Platform configuration header files

src

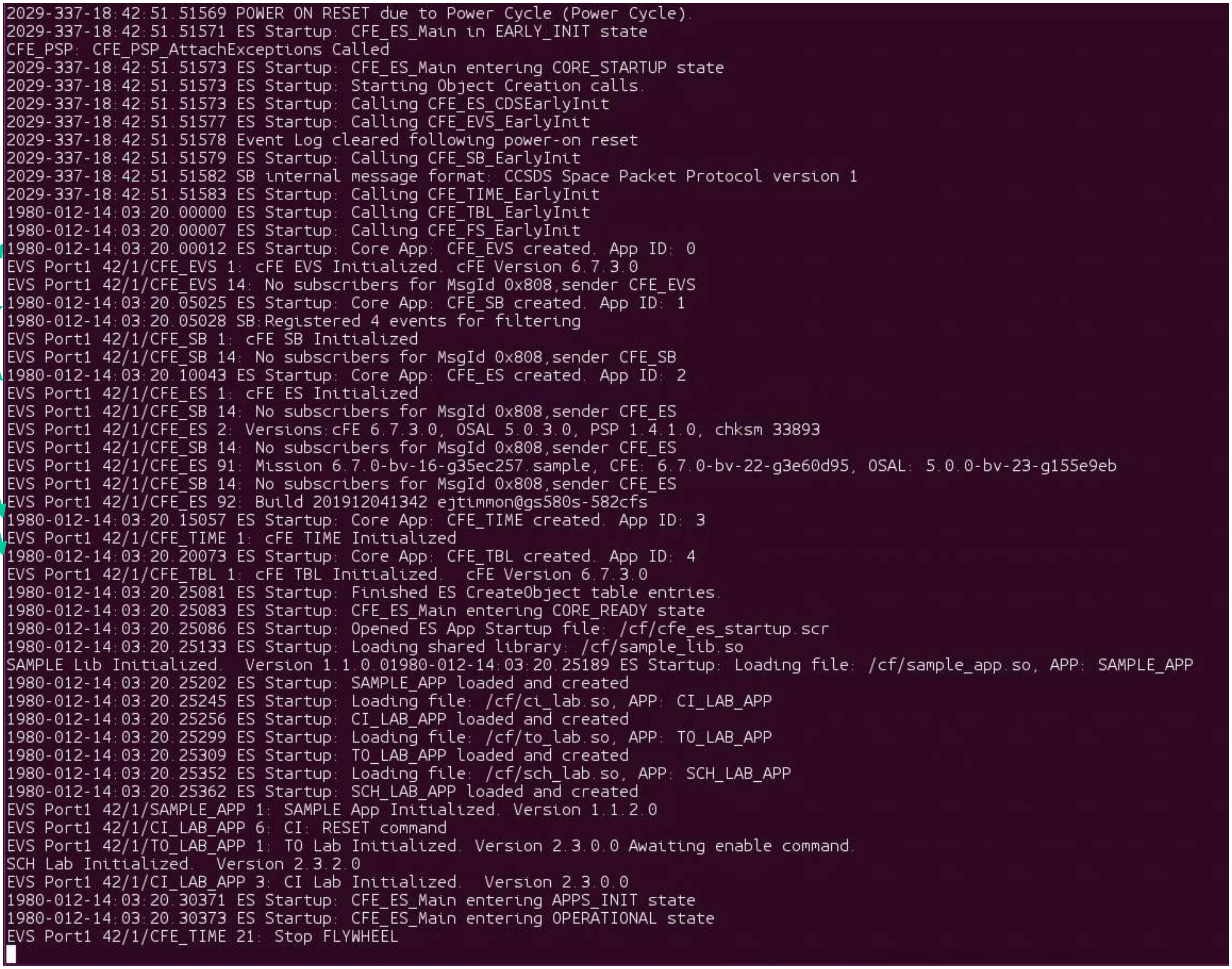
- es
- evs
- fs
- Inc
- make
- sb
- tbl
- time

unit_test

- Procedures
- Results

Exercise 1 Recap

cFE Services Started



```
2029-337-18:42:51.51569 POWER ON RESET due to Power Cycle (Power Cycle).
2029-337-18:42:51.51571 ES Startup: CFE_ES_Main in EARLY_INIT state
CFE_PSP: CFE_PSP_AttachExceptions Called
2029-337-18:42:51.51573 ES Startup: CFE_ES_Main entering CORE_STARTUP state
2029-337-18:42:51.51573 ES Startup: Starting Object Creation calls.
2029-337-18:42:51.51573 ES Startup: Calling CFE_ES_CDSEarlyInit
2029-337-18:42:51.51577 ES Startup: Calling CFE_EVS_EarlyInit
2029-337-18:42:51.51578 Event Log cleared following power-on reset
2029-337-18:42:51.51579 ES Startup: Calling CFE_SB_EarlyInit
2029-337-18:42:51.51582 SB internal message format: CCSDS Space Packet Protocol version 1
2029-337-18:42:51.51583 ES Startup: Calling CFE_TIME_EarlyInit
1980-012-14:03:20.00000 ES Startup: Calling CFE_TBL_EarlyInit
1980-012-14:03:20.00007 ES Startup: Calling CFE_FS_EarlyInit
1980-012-14:03:20.00012 ES Startup: Core App: CFE_EVS created. App ID: 0
EVS Port1 42/1/CFE_EVS 1: cFE EVS Initialized. cFE Version 6.7.3.0
EVS Port1 42/1/CFE_EVS 14: No subscribers for MsgId 0x808,sender CFE_EVS
1980-012-14:03:20.05025 ES Startup: Core App: CFE_SB created. App ID: 1
1980-012-14:03:20.05028 SB:Registered 4 events for filtering
EVS Port1 42/1/CFE_SB 1: cFE SB Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_SB
1980-012-14:03:20.10043 ES Startup: Core App: CFE_ES created. App ID: 2
EVS Port1 42/1/CFE_ES 1: cFE ES Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 2: Versions:cFE 6.7.3.0, OSAL 5.0.3.0, PSP 1.4.1.0, chksm 33893
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 91: Mission 6.7.0-bv-16-g35ec257.sample, CFE: 6.7.0-bv-22-g3e60d95, OSAL: 5.0.0-bv-23-g155e9eb
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 92: Build 201912041342 ejtimmon@gs580s-582cfs
1980-012-14:03:20.15057 ES Startup: Core App: CFE_TIME created. App ID: 3
EVS Port1 42/1/CFE_TIME 1: cFE TIME Initialized
1980-012-14:03:20.20073 ES Startup: Core App: CFE_TBL created. App ID: 4
EVS Port1 42/1/CFE_TBL 1: cFE TBL Initialized.  cFE Version 6.7.3.0
1980-012-14:03:20.25081 ES Startup: Finished ES CreateObject table entries.
1980-012-14:03:20.25083 ES Startup: CFE_ES_Main entering CORE_READY state
1980-012-14:03:20.25086 ES Startup: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.25133 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized.  Version 1.1.0.01980-012-14:03:20.25189 ES Startup: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25202 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25245 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25256 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25299 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25309 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25352 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
1980-012-14:03:20.25362 ES Startup: SCH_LAB_APP loaded and created
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable command.
SCH Lab Initialized.  Version 2.3.2.0
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
1980-012-14:03:20.30371 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30373 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
█
```

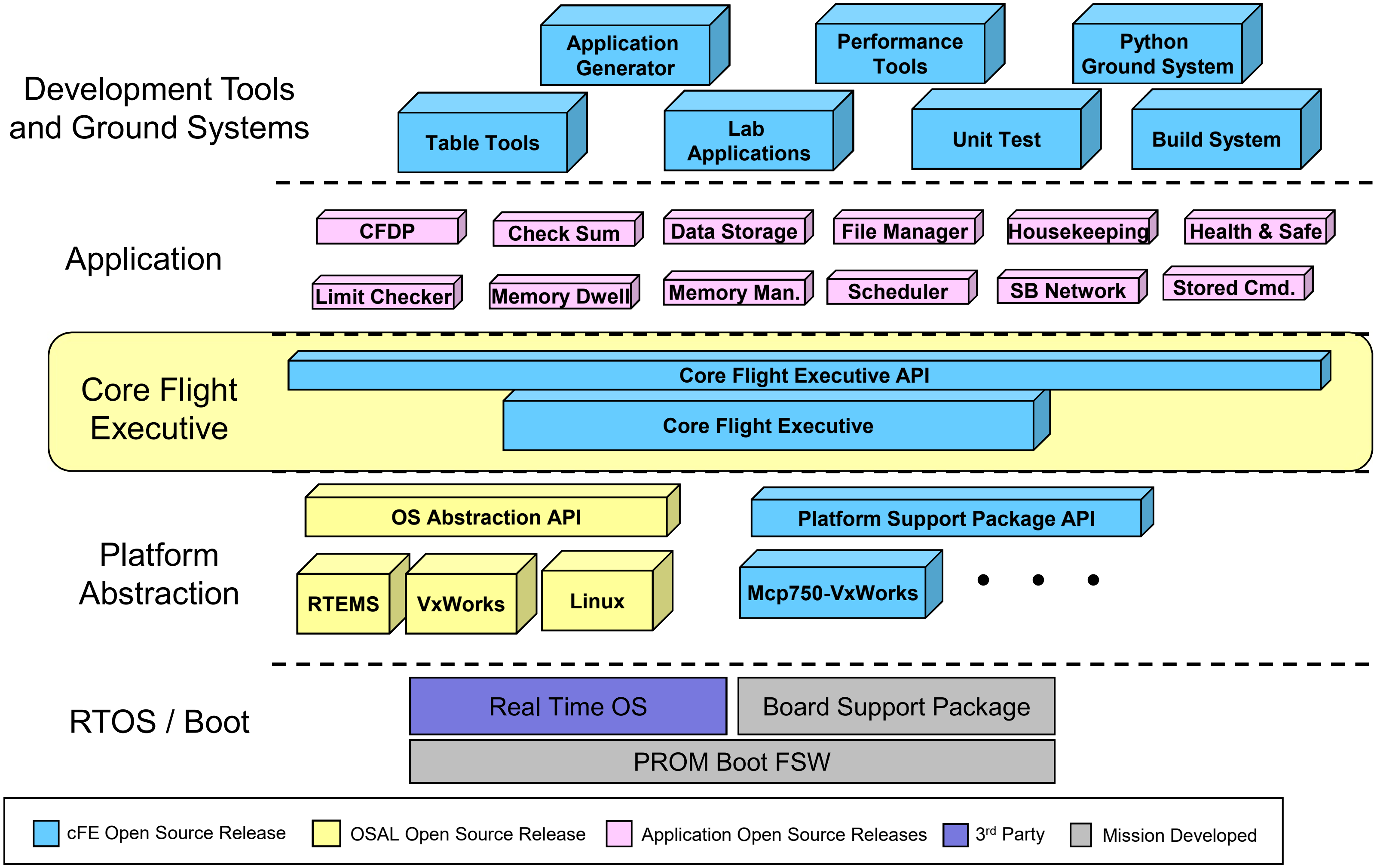
Core Flight System (cFS) Training

Module 2a: Executive Services

Course Agenda

1. **Introduction**
2. **cFE Services**
   a) Executive Services
   b) Time Services
   c) Event Services
   d) Software Bus
   e) Table Services
3. **Application Layer**
   a) cFS Applications
   b) cFS Libraries

Executive Services - cFS Context



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

Board Support Package

PROM Boot FSW

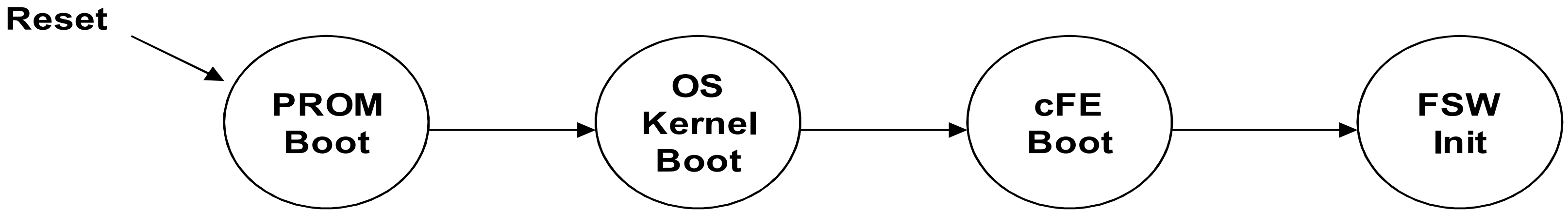
🟦 cFE Open Source Release
🟨 OSAL Open Source Release
▢ Application Open Source Releases
🟪 3rd Party
⬜ Mission Developed

Executive Services (ES) – Overview

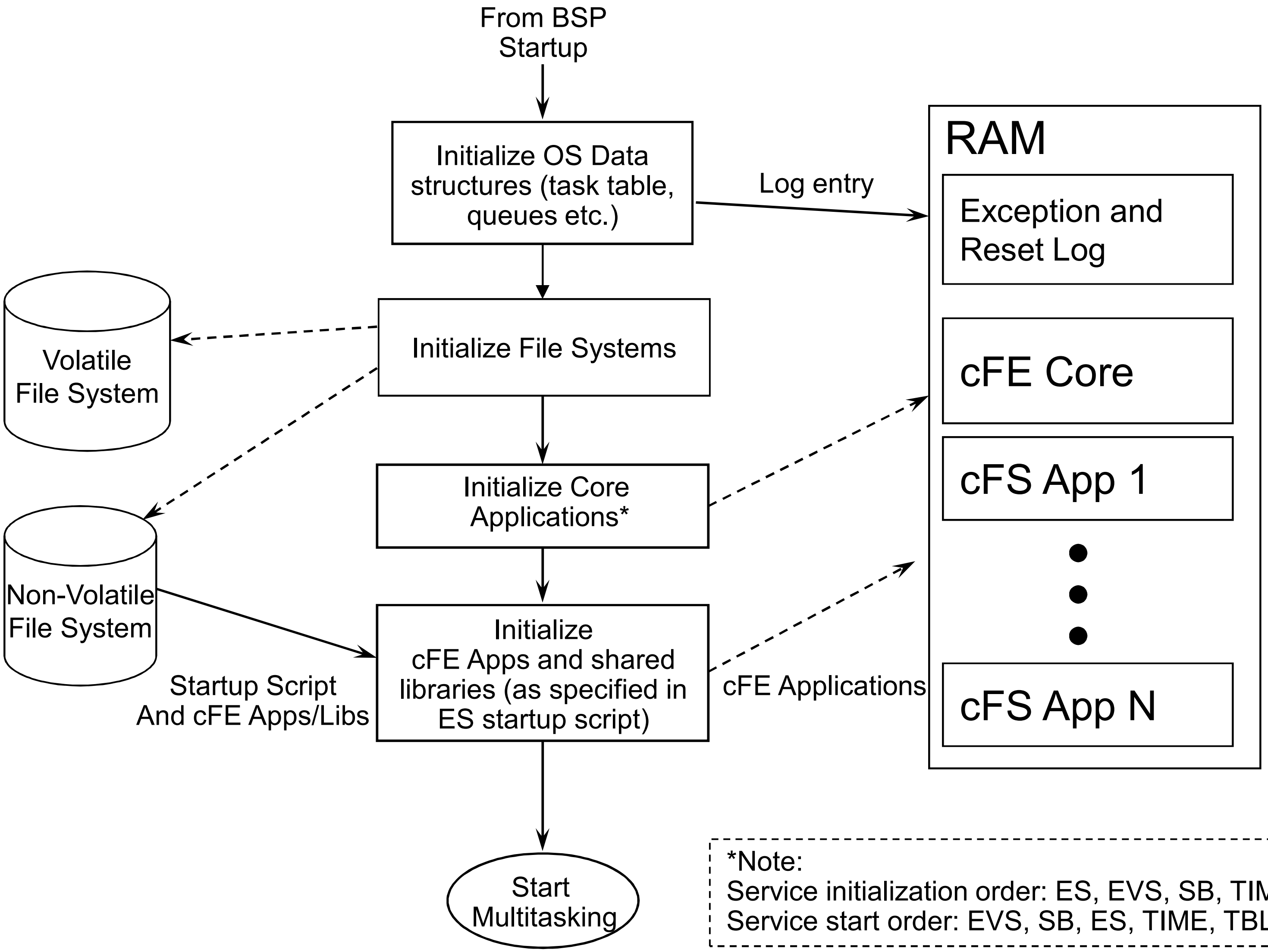
- **Initializes the cFE**
  - Reports reset type
  - Maintains an exception-reset log across processor resets
- **Creates the application runtime environment**
  - Primary interface to underlying operating system task services
  - Manages application resources
  - Starts initial applications according to `cfe_es_startup.scr`
  - Supports starting, stopping, and loading applications during runtime
- **Manages Memory**
  - Provides a dynamic memory pool service
  - Provides Critical Data Stores (CDSs) that are preserved across processor resets

Executive Services - Boot Sequence

- The PROM boots the OS kernel linked with the BSP, loader and EEPROM file system.
  - Accesses simple file system
  - Selects primary and secondary images based on flags and checksum validation
  - Copies OS image to RAM
- The OS kernel boots the cFE
  - Performs self – decompression (optional)
  - Attaches to EEPROM File System
  - Starts up cFE
- cFE boots cFE interface apps and mission components (C&DH, GNC, Science applications)
  - Creates/Attaches to Critical Data Store (CDS)
  - Creates/Attaches to RAM File System
  - Starts cFE applications (EVS, TBL, SB, & TIME)
  - Starts the C&DH and GNC applications based on `cfe_es_startup.scr`



Executive Services - Startup

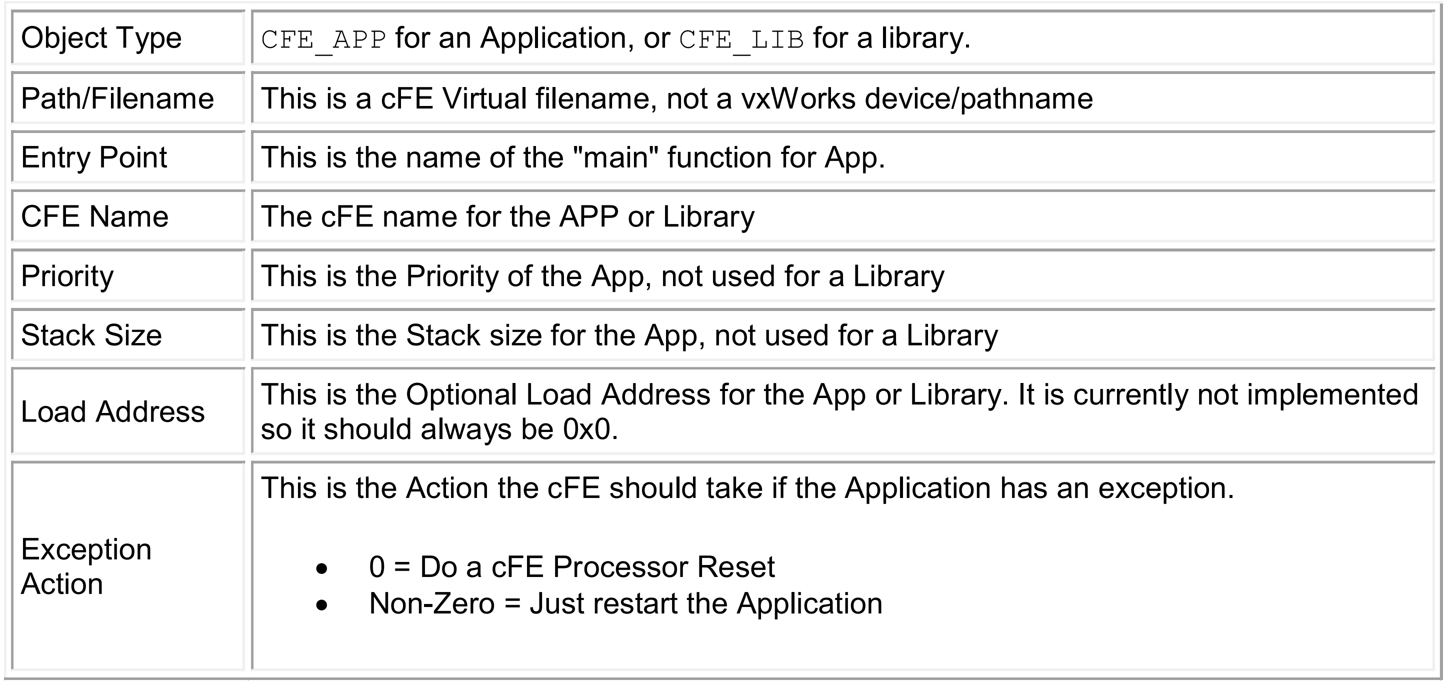

From BSP Startup
Initialize OS Data structures (task table, queues etc.)
Log entry
RAM
Exception and Reset Log
Initialize File Systems
Volatile File System
Non-Volatile File System
cFE Core
cFS App 1
Initialize Core Applications*
Initialize cFE Apps and shared libraries (as specified in ES startup script)
Startup Script And cFE Apps/Libs
cFE Applications
cFS App N
Start Multitasking

*Note:
Service initialization order: ES, EVS, SB, TIME, TBL
Service start order: EVS, SB, ES, TIME, TBL

The cFE core is started as one unit. The cFE Core is linked with the RTOS and support libraries and loaded into system EEPROM as a static executable.

Executive Services - Startup Script

- **The startup script is a text file, written by the user that contains a list of entries (one entry for each application)**
  - Used by the ES application for automating the startup of applications.
  - ES application allows the use of a volatile and nonvolatile startup scripts. The project may utilize zero, one or two startup scripts.



Object Type	`CFE_APP` for an Application, or `CFE_LIB` for a library.
Path/Filename	This is a cFE Virtual filename, not a vxWorks device/pathname
Entry Point	This is the name of the "main" function for App.
CFE Name	The cFE name for the APP or Library
Priority	This is the Priority of the App, not used for a Library
Stack Size	This is the Stack size for the App, not used for a Library
Load Address	This is the Optional Load Address for the App or Library. It is currently not implemented so it should always be 0x0.
Exception Action	This is the Action the cFE should take if the Application has an exception. • 0 = Do a cFE Processor Reset • Non-Zero = Just restart the Application

Executive Services – Example Script

```
CFE_APP, /cf/apps/ci_lab.o,   CI_Lab_AppMain,  CI_LAB_APP,     70,   4096, 0x0, 0;
CFE_APP, /cf/apps/sch_lab.o,  SCH_Lab_AppMain, SCH_LAB_APP,   120,   4096, 0x0, 0;
CFE_APP, /cf/apps/to_lab.o,   TO_Lab_AppMain,  TO_LAB_APP,     74,   4096, 0x0, 0;
CFE_LIB, /cf/apps/cfs_lib.o,  CFS_LibInit,     CFS_LIB,         0,      0, 0x0, 0;
!
! Startup script fields:
! 1. Object Type      -- CFE_APP for an Application, or CFE_LIB for a library.
! 2. Path/Filename    -- This is a cFE Virtual filename, not a vxWorks device/pathname
! 3. Entry Point      -- This is the "main" function for Apps.
! 4. CFE Name         -- The cFE name for the the APP or Library
! 5. Priority         -- This is the Priority of the App, not used for Library
! 6. Stack Size       -- This is the Stack size for the App, not used for the Library
! 7. Load Address     -- This is the Optional Load Address for the App or Library. Currently
not implemented
!                        so keep it at 0x0.
! 8. Exception Action -- This is the Action the cFE should take if the App has an exception.
!                        0        = Just restart the Application
!                        Non-Zero = Do a cFE Processor Reset
!
! Other  Notes:
! 1. The software will not try to parse anything after the first '!' character it sees. That
!    is the End of File marker.
! 2. Common Application file extensions:
!    Linux = .so  ( ci.so )
!    OS X  = .bundle  ( ci.bundle )
!    Cygwin = .dll ( ci.dll )
!    vxWorks = .o ( ci.o )
!    RTEMS with S-record Loader = .s3r ( ci.s3r )
!    RTEMS with CEXP Loader = .o ( ci.o )
```

Executive Services – Logs

- **Exception-Reset**
  - Logs information related to resets and exceptions
- **System Log**
  - cFE apps use this log when errors are encountered during initialization before the Event Services is fully initialized
  - Mission apps can also use it during initialization
    - Recommended that apps should register with event service immediately after registering with ES so app events are captured in the EVS log
  - Implemented as an array of bytes that has variable length strings produced by printf() type statements

Executive Services – Reset Behavior

- **Power-on Reset**
  - Operating system loaded and started prior to cFE
  - Initializes file system
  - Critical data stores and logs cleared (initialized by hardware first)
  - ES starts each cFE service and then the mission applications
- **Processor Reset Preserves**
  - File system
  - Critical Data Store (CDS)
  - ES System Log
  - ES Exception and Reset (ER) log
  - Performance Analysis data
  - ES Reset info (i.e. reset type, boot source, number of processor resets)
  - Time Data (i.e. MET, STCF, Leap Seconds)
- **A power-on reset will be performed after a configurable number of processor resets**
  - Ground responsible for managing processor reset counter

Executive Services – Retrieving Onboard State

- **Telemetry**
  - Housekeeping Status
    - Log file states, App, Resets, Performance Monitor, Heap Stats
- **Telemetry packets generated by command**
  - Single App Information
  - Memory Pool Statistics Packet
  - Shell command output packet
- **Files generated by command**
  - System Log
  - Exception-Reset Log
  - Performance Monitor
  - Critical Data Store Registry
  - All registered apps
  - All registered tasks

Executive Services -
System Integration and App Development (1 of 2)

- **Child Tasks**
  - Recommend creating during app initialization
  - Relative parent priority depends on child’s role
    - Performing lengthy process may be lower
    - Servicing short duration I/O may be higher

OS	Call
POSIX/Linux	pthread_create()
RTEMS	rtems_task_create()
VxWorks	taskSpawn()

Executive Services - System Integration and App Development (2 of 2)

- **Query startup type (Power On vs Processor)**
  - Not commonly used since CDS performs data preservation

- **Critical Data Store (CDS)**
  - E.g. Data Storage maintains open file management data in a CDS
  - Typical code idiom in app’s initialization

```
Result = CFE_ES_RegisterCDS()
if (Result == CFE_SUCCESS)
   Populate CDS
else if (Result == CFE_ES_CDS_ALREADY_EXISTS)
   Restore CDS data
… Continually update CDS as application executes
```

- **Memory Pool**
  - Ideally apps would allocate memory pools during initialization but there aren’t any restrictions
  - cFE Examples: Software Bus, Tables, and Events
  - App Examples: CFDP and Housekeeping

Executive Services - APIs

Memory Pool Functions	Purpose
CFE_ES_PoolCreateNoSem	Initializes a memory pool created by an application without using a semaphore during processing
CFE_ES_PoolCreate	Initializes a memory pool created by an application while using a semaphore during processing
CFE_ES_PoolCreateEx	Initializes a memory pool created by an application with application specified block sizes
CFE_ES_GetPoolBuf	Gets a buffer from the memory pool created by #CFE_ES_PoolCreate or #CFE_ES_PoolCreateNoSem
CFE_ES_GetPoolBufInfo	Gets info on a buffer previously allocated via #CFE_ES_GetPoolBuf
CFE_ES_PutPoolBuf	Releases a buffer from the memory pool that was previously allocated via #CFE_ES_GetPoolBuf
CFE_ES_GetMemPoolStats	Extracts the statistics maintained by the memory pool software

Executive Services - APIs

API List (1 of 2)	Purpose
CFE_ES_GetResetType	Return the most recent Reset Type
CFE_ES_ResetCFE	Reset the cFE Core and all cFE Applications
CFE_ES_RestartApp	Restart a single cFE App
CFE_ES_ReloadApp	Reload a single cFE App
CFE_ES_DeleteApp	Delete a cFE App
CFE_ES_ExitApp	Exit a cFE Application
CFE_ES_RunLoop	Check for Exit, Restart, or Reload commands
CFE_ES_WaitForSystemState	Allow an Application to Wait for a minimum global system state
CFE_ES_WaitForStartupSync	Allow an Application to Wait for the "OPERATIONAL" global system state
CFE_ES_GetAppIDByName	Get an Application ID associated with a specified Application name
CFE_ES_GetAppID	Get an Application ID for the calling Application
CFE_ES_GetAppName	Get an Application name for a specified Application ID
CFE_ES_GetAppInfo	Get Application Information given a specified App ID
CFE_ES_GetTaskInfo	Get Task Information given a specified Task ID

Executive Services - APIs

API List (2 of 2)	Purpose
CFE_ES_CreateChildTask	Creates a new task under an existing Application
CFE_ES_RegisterChildTask	Registers a cFE Child task associated with a cFE Application
CFE_ES_IncrementTaskCounter	Increments the execution counter for the calling task
CFE_ES_DeleteChildTask	Deletes a task under an existing Application
CFE_ES_ExitChildTask	Exits a child task
CFE_ES_WriteToSysLog	Write a string to the cFE System Log
CFE_ES_CalculateCRC	Calculate a CRC on a block of memory
CFE_ES_RegisterCDS	Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
CFE_ES_CopyToCDS	Save a block of data in the Critical Data Store (CDS)
CFE_ES_RestoreFromCDS	Recover a block of data from the Critical Data Store (CDS)
CFE_ES_RegisterGenCounter	Register a generic counter
CFE_ES_DeleteGenCounter	Delete a generic counter
CFE_ES_IncrementGenCounter	Increments the specified generic counter
CFE_ES_SetGenCount	Set the specified generic counter
CFE_ES_GetGenCount	Get the specified generic counter count
CFE_ES_GetGenCounterIDByName	Get the Id associated with a generic counter name
CFE_ES_ProcessCoreException	Process an exception detected by the underlying OS/PSP

Executive Services – Command List

Command List	Purpose
CFE_ES_StartPerfDataCmd	Start performance data
CFE_ES_StopPerfDataCmd	Stop performance data
CFE_ES_SetPerfFilterMaskCmd	Set performance filter mask
CFE_ES_SetPerfTriggerMaskCmd	Set performance trigger mask
CFE_ES_HousekeepingCmd	On-board command (HK request)
CFE_ES_NoopCmd	ES task ground command (NO-OP)
CFE_ES_ResetCountersCmd	ES task ground command (reset counters)
CFE_ES_RestartCmd	Restart cFE (may reset processor)
CFE_ES_ShellCmd	Pass thru string to O/S shell
CFE_ES_StartAppCmd	Load (and start) single application
CFE_ES_StopAppCmd	Stop single application
CFE_ES_RestartAppCmd	Restart a single application
CFE_ES_ReloadAppCmd	Reload a single application
CFE_ES_QueryOneCmd	Request tlm packet with single app data
CFE_ES_QueryAllCmd	Write all app data to file
CFE_ES_QueryAllTasksCmd	Write all Task Data to a file
CFE_ES_ClearSyslogCmd	Clear executive services system log
CFE_ES_OverWriteSyslogCmd	Set syslog mode
CFE_ES_WriteSyslogCmd	Process Cmd to write ES System Log to file
CFE_ES_ClearERLogCmd	Clear The exception and reset log
CFE_ES_WriteERLogCmd	Process Cmd to write exception & reset log to a file
CFE_ES_VerifyCmdLength	Verify command packet length
CFE_ES_ResetPRCountCmd	ES task ground command (Processor Reset Count)
CFE_ES_SetMaxPRCountCmd	Set Maximum Processor reset count
CFE_ES_DeleteCDSCmd	Delete Specified Critical Data Store
CFE_ES_SendMemPoolStatsCmd	Telemeter Memory Pool Statistics
CFE_ES_DumpCDSRegistryCmd	Dump CDS Registry to a file

Executive Services – Configuration Parameters

Command List	Purpose
CFE_PLATFORM_ES_MAX_APPLICATIONS	Max Number of Applications
CFE_PLATFORM_ES_MAX_LIBRARIES	Max Number of Shared libraries
CFE_PLATFORM_ES_ER_LOG_ENTRIES	Max Number of ER (Exception and Reset) log entries
CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE	Maximum size of CPU Context in ES Error Log
CFE_PLATFORM_ES_SYSTEM_LOG_SIZE	Size of the cFE System Log
CFE_PLATFORM_ES_OBJECT_TABLE_SIZE	Number of entries in the ES Object table
CFE_PLATFORM_ES_MAX_GEN_COUNTERS	Max Number of Generic Counters
CFE_PLATFORM_ES_APP_SCAN_RATE	ES Application Control Scan Rate
CFE_PLATFORM_ES_APP_KILL_TIMEOUT	ES Application Kill Timeout
CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE	ES Ram Disk Sector Size
CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS	ES Ram Disk Number of Sectors
CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED	Percentage of Ram Disk Reserved for Decompressing Apps
CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING	RAM Disk Mount string
CFE_PLATFORM_ES_CDS_SIZE	Critical Data Store Size
CFE_PLATFORM_ES_USER_RESERVED_SIZE	User Reserved Memory Size
CFE_PLATFORM_ES_RESET_AREA_SIZE	ES Reset Area Size
CFE_PLATFORM_ES_NONVOL_STARTUP_FILE	ES Nonvolatile Startup Filename
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE	ES Volatile Startup Filename
CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME	Default Shell Filename
CFE_PLATFORM_ES_MAX_SHELL_CMD	Max Shell Command Size
CFE_PLATFORM_ES_MAX_SHELL_PKT	Shell Command Telemetry Pkt Segment Size
CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE	Default Application Information Filename
CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE	Default Application Task Information Filename
CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE	Default System Log Filename
CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE	Default Exception and Reset (ER) Log Filename

Executive Services – Configuration Parameters

Command List	Purpose
CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME	Default Performance Data Filename
CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE	Default Critical Data Store Registry Filename
CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE	Default System Log Mode
CFE_PLATFORM_ES_PERF_MAX_IDS	Max Number of Performance IDs
CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE	Max Size of Performance Data Buffer
CFE_PLATFORM_ES_PERF_FILTMASK_NONE	Filter Mask Setting for Disabling All Performance Entries
CFE_PLATFORM_ES_PERF_FILTMASK_ALL	Filter Mask Setting for Enabling All Performance Entries
CFE_PLATFORM_ES_PERF_FILTMASK_INIT	Default Filter Mask Setting for Performance Data Buffer
CFE_PLATFORM_ES_PERF_TRIGMASK_NONE	Default Filter Trigger Setting for Disabling All Performance Entries
CFE_PLATFORM_ES_PERF_TRIGMASK_ALL	Filter Trigger Setting for Enabling All Performance Entries
CFE_PLATFORM_ES_PERF_TRIGMASK_INIT	Default Filter Trigger Setting for Performance Data Buffer
CFE_PLATFORM_ES_PERF_CHILD_PRIORITY	Performance Analyzer Child Task Priority
CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE	Performance Analyzer Child Task Stack Size
CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY	Performance Analyzer Child Task Delay
CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS	Performance Analyzer Child Task Number of Entries Between Delay
CFE_PLATFORM_ES_DEFAULT_STACK_SIZE	Default Stack Size for an Application
CFE_PLATFORM_ES_EXCEPTION_FUNCTION	cFE Core Exception Function
CFE_PLATFORM_ES_START_TASK_PRIORITY	ES Task Priority
CFE_PLATFORM_ES_START_TASK_STACK_SIZE	ES Task Stack Size
CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES	Maximum Number of Registered CDS Blocks
CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS	Number of Processor Resets Before a Power On Reset
CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE	ES Critical Data Store Max Memory Pool Block Size
CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC	Poll timer for startup sync delay
CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC	Startup script timeout

Exercise 2 - Command cFE Executive Service

Part 1 – Start the Ground System

The cFS-GroundSystem tool can be used to send commands and receive telemetry (see https://github.com/nasa/cFS-GroundSystem/tree/master/Guide-GroundSystem.txt, the Guide-GroundSystem.txt). Note it depends on PyQt4 and PyZMQ:

1. Ensure that cFE is running
2. Open a new terminal
3. Compile cmdUtil and start the ground system executable

```
cd cFS/tools/cFS-GroundSystem/Subsystems/cmdUtil
make
cd ../..
python GroundSystem.py
```

4. Select "Start Command System"
5. Select "Enable Tlm"
6. Enter IP address of system executing cFS (127.0.0.1 if running locally) into the "Input" field and click "Send"
7. In the original ground system window, select "Start Telemetry System"

At this point, telemetry should be visible in the ground system

Exercise 2 - Command cFE Executive Service

Part 2 – Command Executive Services

8. On the Command System Main Page, select "ES No-Op".

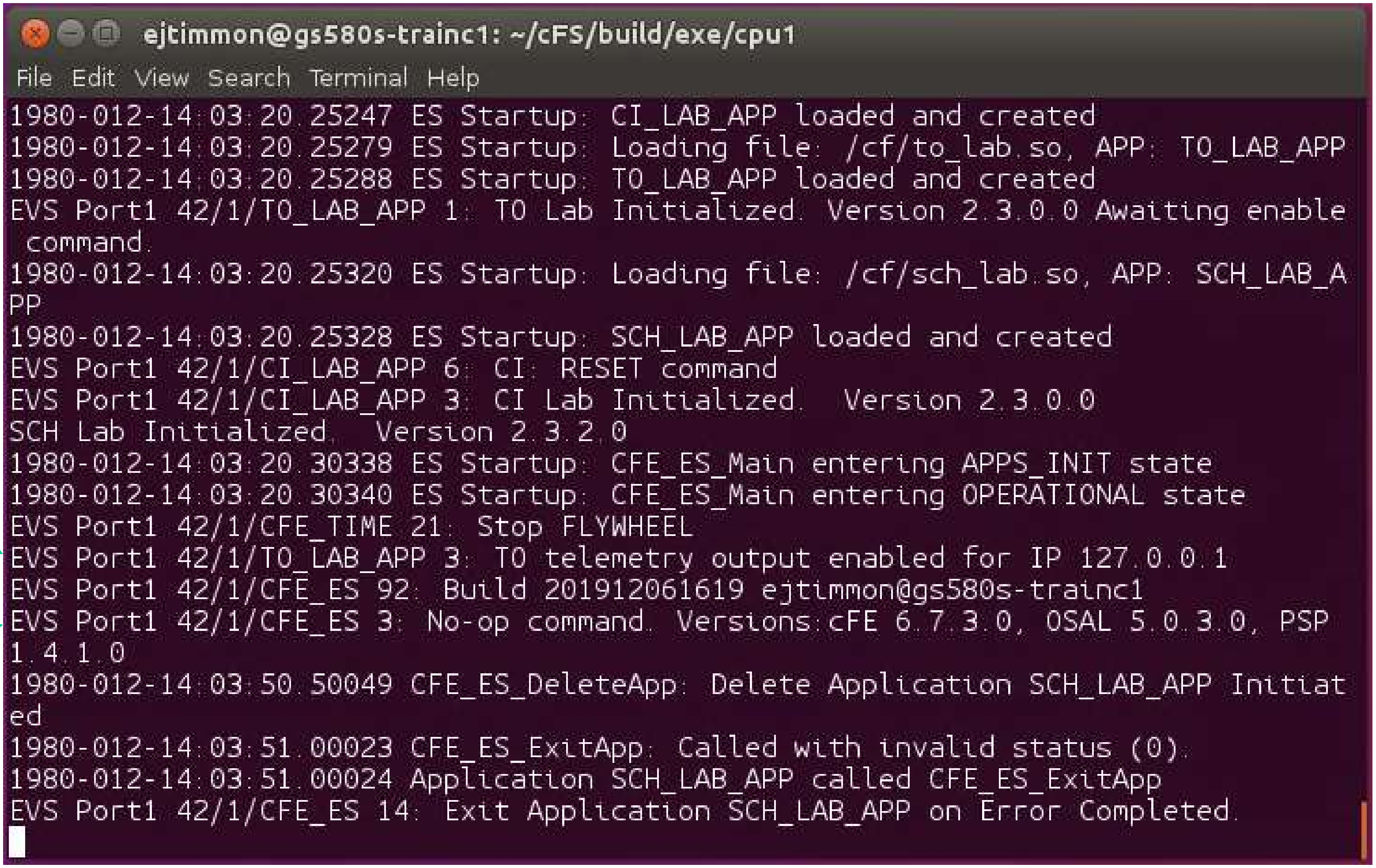
- A no-op message should appear in the cFS screen.

9. Reload an application.

- On the Command System Main Page, click the "Display Page" button beside "Executive Services".
- Click the "Send" button beside "Stop and Unload Application".
- Enter "SCH_LAB_APP" in the "Input" field.
- Click "Send".

NOTE: "SCH_LAB_APP" is the cFE name specified for one of the apps in the cfe_es_startup.scr file. Many cFE ES commands require the cFE name of an application or library as a parameter

Exercise 2 Recap


ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1
File Edit View Search Terminal Help

```
1980-012-14:03:20.25247 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25279 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25288 ES Startup: TO_LAB_APP loaded and created
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable
 command.
1980-012-14:03:20.25320 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_A
PP
1980-012-14:03:20.25328 ES Startup: SCH_LAB_APP loaded and created
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
SCH Lab Initialized.  Version 2.3.2.0
1980-012-14:03:20.30338 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30340 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 42/1/CFE_ES 92: Build 201912061619 ejtimmon@gs580s-trainc1
EVS Port1 42/1/CFE_ES 3: No-op command. Versions:cFE 6.7.3.0, OSAL 5.0.3.0, PSP
1.4.1.0
1980-012-14:03:50.50049 CFE_ES_DeleteApp: Delete Application SCH_LAB_APP Initiat
ed
1980-012-14:03:51.00023 CFE_ES_ExitApp: Called with invalid status (0).
1980-012-14:03:51.00024 Application SCH_LAB_APP called CFE_ES_ExitApp
EVS Port1 42/1/CFE_ES 14: Exit Application SCH_LAB_APP on Error Completed.
```

Enable Tlm Command

ES No-Op Command

ES Delete App Command

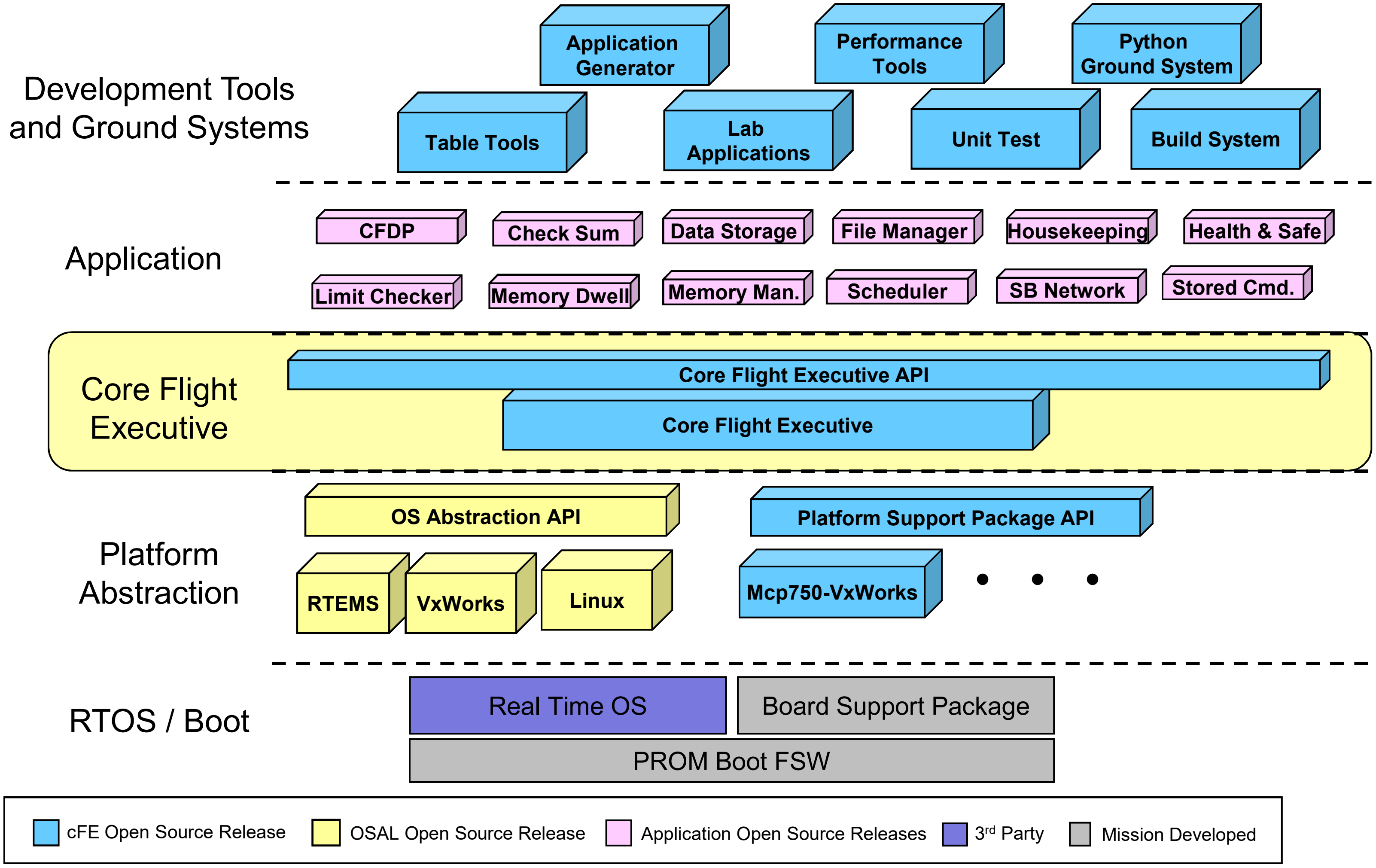
Core Flight System (cFS) Training

Module 2b: Time Services

Course Agenda

1. **Introduction**
2. **cFE Services**
   a) Executive Services
   b) Time Services
   c) Event Services
   d) Software Bus
   e) Table Services
3. **Application Layer**
   a) cFS Applications
   b) cFS Libraries

Time Service - cFS Context



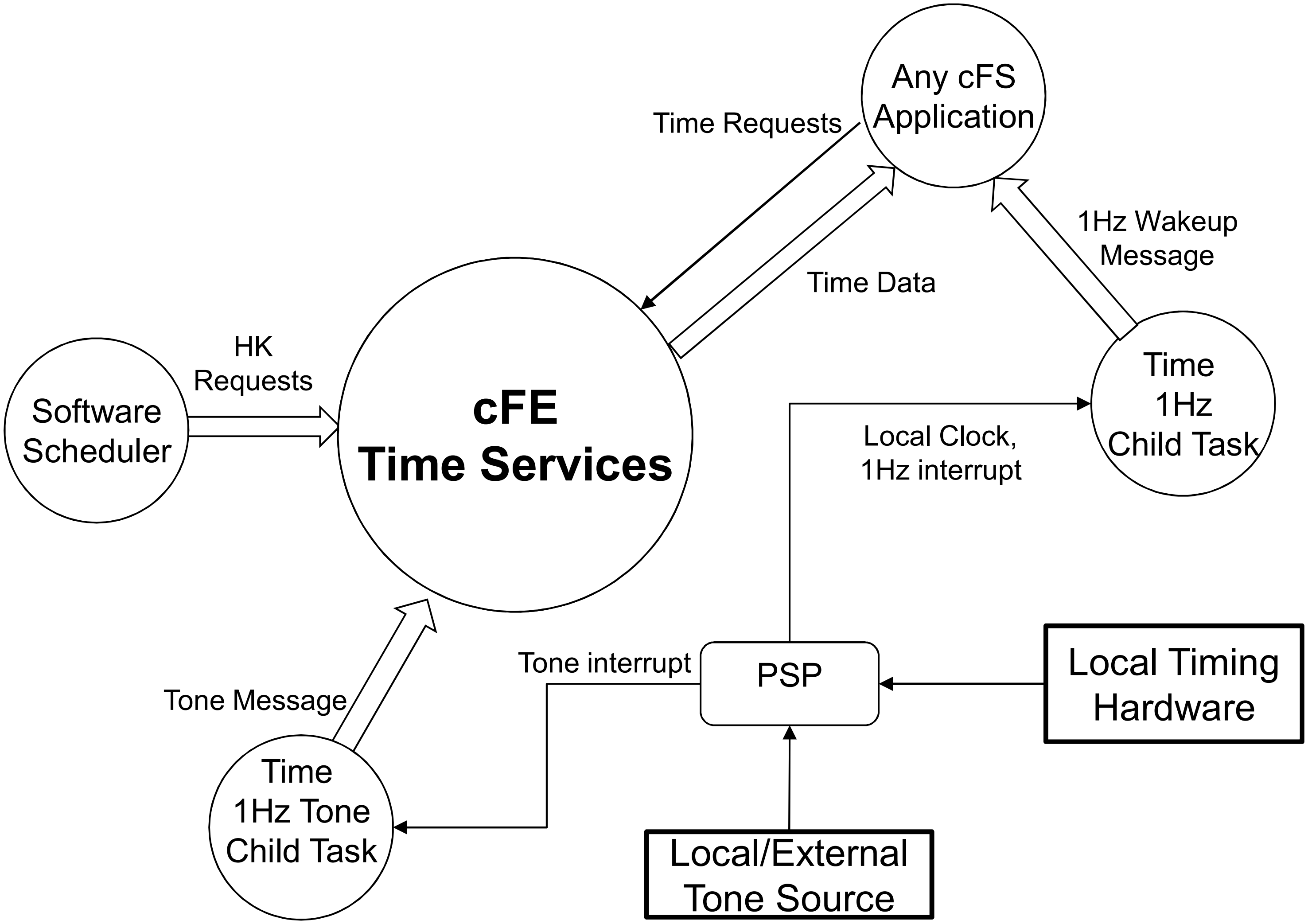
Time Services - Overview

- Provides time correlation, distribution and synchronization services
- Provides a user interface for correlation of spacecraft time to the ground reference time (epoch)
- Provides calculation of spacecraft time, derived from mission elapsed time (MET), a spacecraft time correlation factor (STCF), and optionally, leap seconds
- Provides a functional API for cFE applications to query the time
- Distributes a “time at the tone” command packet, containing the correct time at the moment of the 1Hz tone signal
- Distributes a “1Hz wakeup” command packet
- Forwards tone and time-at-the-tone packets
- Designing and configuring time is tightly coupled with the mission avionics design

Time Services – Time Formats

- **Supports two formats**
- **International Atomic Time (TAI)**
  - Number of seconds and sub-seconds elapsed since the ground epoch
  - TAI = MET + STCF
    - Mission Elapsed Counter (MET) time since powering on the hardware containing the counter
    - Spacecraft Time Correlation Factor (STCF) set by ground ops
    - Note STCF can correlate MET to any time epoch so TAI is mandated
- **Coordinated Universal Time (UTC)**
  - Synchronizes time with astronomical observations
  - UTC = TAI – Leap Seconds
  - Leap Seconds account for earth’s slowing rotation

Time Services - Context



Time Services – “Flywheeling”

- ***Flywheeling*** **occurs when TIME is not getting a valid tone signal or external "time at the tone" message. While this has minimal impact on internal operations, it can result in the drifting apart of times being stored by different spacecraft systems.**

- **Flywheeling occurs when at least one of the following conditions is true:**
  - loss of tone signal
  - loss of "time at the tone" data packet
  - signal and packet not within valid window
  - commanded into fly-wheel mode

Time Services – Reset Behavior

- **Power-On-Reset**
  - Initializes all counters in housekeeping telemetry
  - Validity state set to Invalid
  - STCF, Leap Seconds, and 1 Hz Adjustment to zero set to zero

- **Processor reset, preserves:**
  - MET
  - STCF
  - Leap Seconds
  - Clock Signal Selection
  - Current Time Client Delay (if applicable)
  - Uses ‘signature’ to determine validity of saved time. If signature fails then power-on-reset initialization is performed

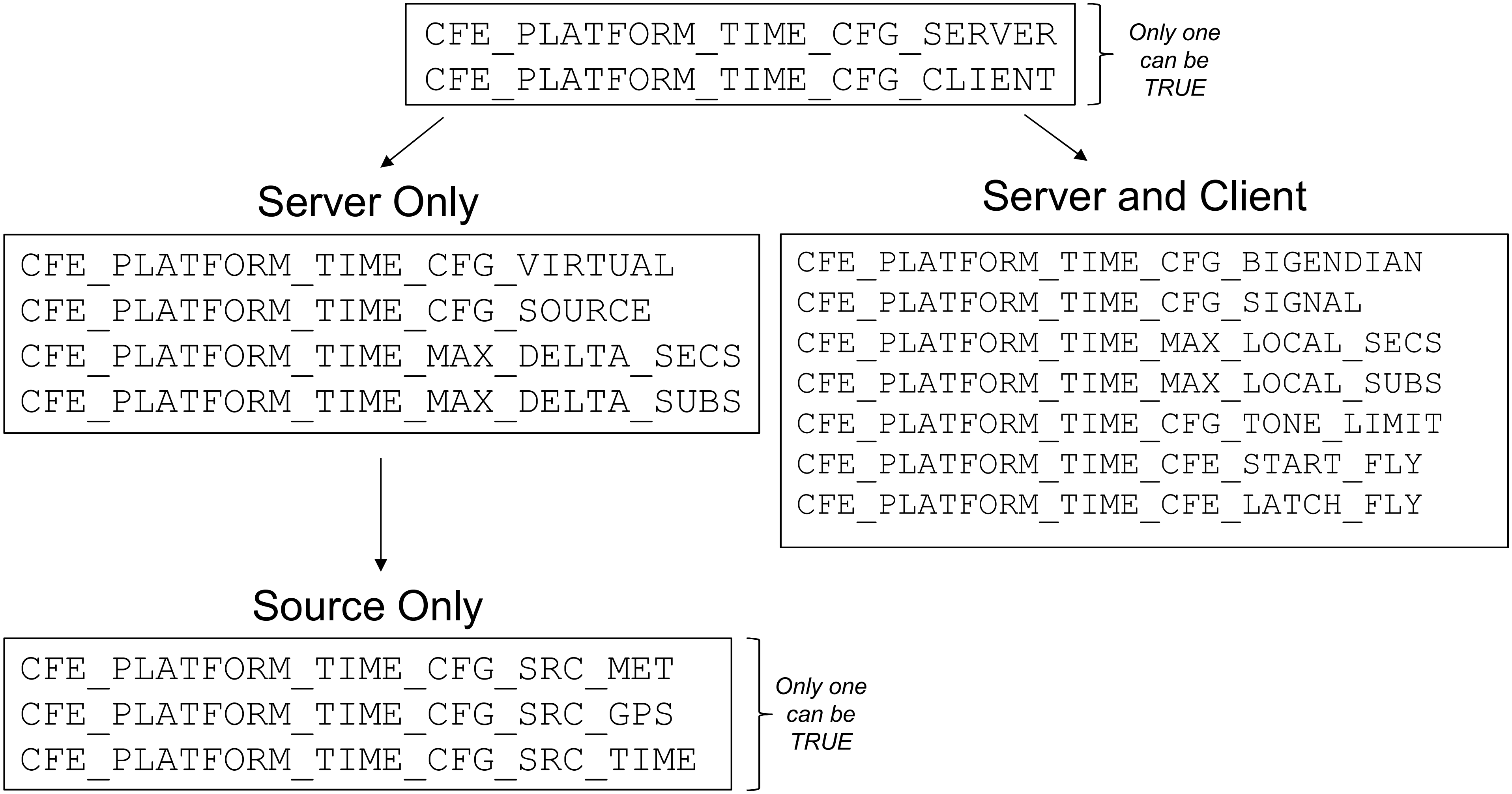
Time Services – Retrieving Onboard State

- **Telemetry**
  - Housekeeping Status
    - Clock state, Leap Seconds, MET, STCF 1Hz Adjust
- **Telemetry packets generated by command**
  - Diagnostic Packet
- **Files generated by command**
  - None

Time Services – Configuration Considerations

- What is your time format?
- Are you setting time or receiving time?
- Is your MET provided by local hardware?
- Is time coming from an external source?
- How long can you go without synchronizing time?

Time Services – Configuration Parameters



```
CFE_PLATFORM_TIME_CFG_SERVER
CFE_PLATFORM_TIME_CFG_CLIENT
```

Only one can be TRUE

Server Only

```
CFE_PLATFORM_TIME_CFG_VIRTUAL
CFE_PLATFORM_TIME_CFG_SOURCE
CFE_PLATFORM_TIME_MAX_DELTA_SECS
CFE_PLATFORM_TIME_MAX_DELTA_SUBS
```

Server and Client

```
CFE_PLATFORM_TIME_CFG_BIGENDIAN
CFE_PLATFORM_TIME_CFG_SIGNAL
CFE_PLATFORM_TIME_MAX_LOCAL_SECS
CFE_PLATFORM_TIME_MAX_LOCAL_SUBS
CFE_PLATFORM_TIME_CFG_TONE_LIMIT
CFE_PLATFORM_TIME_CFE_START_FLY
CFE_PLATFORM_TIME_CFE_LATCH_FLY
```

Source Only

```
CFE_PLATFORM_TIME_CFG_SRC_MET
CFE_PLATFORM_TIME_CFG_SRC_GPS
CFE_PLATFORM_TIME_CFG_SRC_TIME
```

Only one can be TRUE

Time Services - APIs

Basic Clock Functions	Purpose
CFE_TIME_GetTime	Get the current spacecraft time
CFE_TIME_GetUTC	Get the current UTC time
CFE_TIME_GetTAI	Get the current TAI time
CFE_TIME_MET2SCTIME	Converts MET to Spacecraft time
CFE_TIME_GetMET	Get the current value of the mission-elapsed time
CFE_TIME_GetMETseconds	Get the current seconds count of the mission-elapsed time
CFE_TIME_GetMETsubsecs	Get the current sub-seconds count of the mission-elapsed time
CFE_TIME_GetSTCF	Get the current value of the spacecraft time correction factor (STCF)
CFE_TIME_GetLeapSeconds	Get the current value of the leap seconds counter
CFE_TIME_GetClockState	Get the current state of the spacecraft clock
CFE_TIME_GetClockInfo	Get clock information
CFE_TIME_Compare	Compare two CFE_TIME_SysTime_t values
CFE_TIME_Print	Create text string representing date and time
CFE_TIME_RegisterSynchCallback	Register synch callback function
CFE_TIME_UnregisterSynchCallback	Unregister synch callback function

Time Services - APIs

Time Conversion Functions	Purpose
CFE_TIME_Sub2MicroSecs	Convert a sub-seconds count to an equivalent number of microseconds
CFE_TIME_Micro2SubSecs	Convert a number of microseconds to an equivalent sub-seconds count
CFE_TIME_CFE2FSSeconds	Convert cFE seconds to File System Seconds
CFE_TIME_FS2CFESeconds	Convert File System seconds to cFE seconds

Time Manipulation Functions	Purpose
CFE_TIME_Add	Add two time values
CFE_TIME_Subtract	Subtract one time value from another

External Time Sources	Purpose
CFE_TIME_ExternalTone	Latch the local time at the 1Hz tone signal
CFE_TIME_ExternalMET	Provide the MET from an external source
CFE_TIME_ExternalGPS	Provide the time from an external source that has data common to GPS receiver
CFE_TIME_ExternalTime	Provide the time from an external source that measures time relative to a known epoch

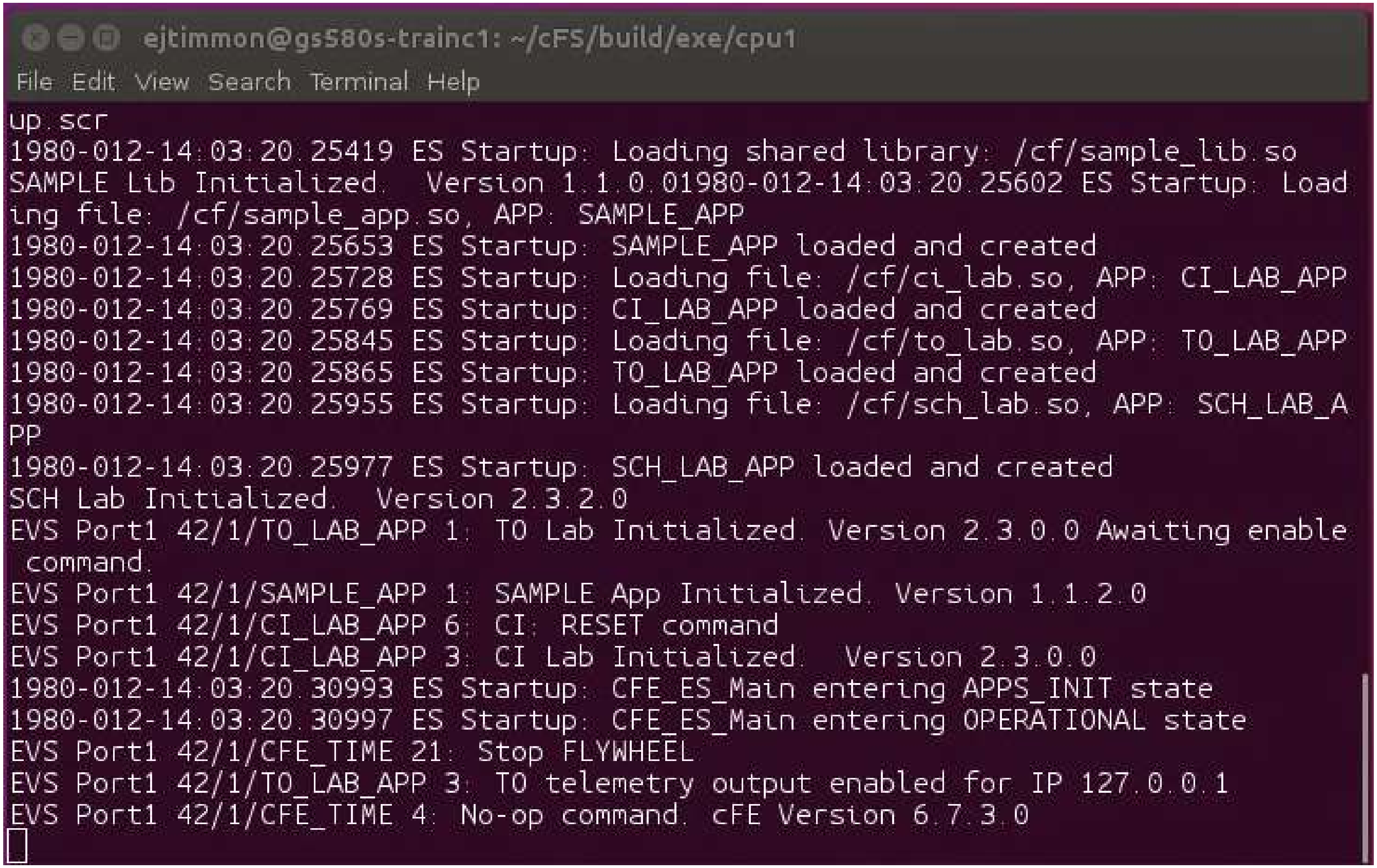
Time Services Commands

Command Functions	Purpose
CFE_TIME_Add1HZAdjustmentCmd	Time task ground command (1Hz adjust: Add)
CFE_TIME_AddAdjustCmd	Time task ground command (Add delta adjust)
CFE_TIME_AddDelayCmd	Time task ground command (add tone delay)
CFE_TIME_SendDiagnosticTlm	Time task ground command (diagnostics)
CFE_TIME_NoopCmd	Time task ground command (NO-OP)
CFE_TIME_ResetCountersCmd	Time task ground command (reset counters)
CFE_TIME_SetLeapSecondsCmd	Time task ground command (set leaps)
CFE_TIME_SetMETCmd	Time task ground command (set MET)
CFE_TIME_SetSignalCmd	Time task command (primary/redundant tone signal selection)
CFE_TIME_SetSourceCmd	Time task command (set time source)
CFE_TIME_SetStateCmd	Time task command (set clock state)
CFE_TIME_SetSTCFCmd	Time task ground command (set STCF [time server only])
CFE_TIME_SetTimeCmd	Time task ground command (Basically sets STCF...but if time format is UTC, removes leap seconds [should also be time server only])
CFE_TIME_Sub1HZAdjustmentCmd	Time task ground command (1Hz adjust: Subtract)
CFE_TIME_SubAdjustCmd	Time task ground command (Subtract delta adjust)
CFE_TIME_SubDelayCmd	Time task ground command (subtract tone delay)

Exercise 3 - Command cFE Time Service

1. Ensure that cFE is running
2. Open a new terminal
3. Start the ground system executable (as in Exercise 2)
4. Select "Start Command System"
5. Select "Enable Tlm"
6. Enter IP address of system executing cFS (127.0.0.1 if running locally) into the "Input" field and click "Send"
7. Select "Time No-Op"
    - Click "Send"

Exercise 3 Recap


ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1
File Edit View Search Terminal Help

```
up.scr
1980-012-14:03:20.25419 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized.  Version 1.1.0.01980-012-14:03:20.25602 ES Startup: Load
ing file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25653 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25728 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25769 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25845 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25865 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25955 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_A
PP
1980-012-14:03:20.25977 ES Startup: SCH_LAB_APP loaded and created
SCH Lab Initialized.  Version 2.3.2.0
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable
 command.
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
1980-012-14:03:20.30993 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30997 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 42/1/CFE_TIME 4: No-op command. cFE Version 6.7.3.0
```

TIME
No-Op
Command

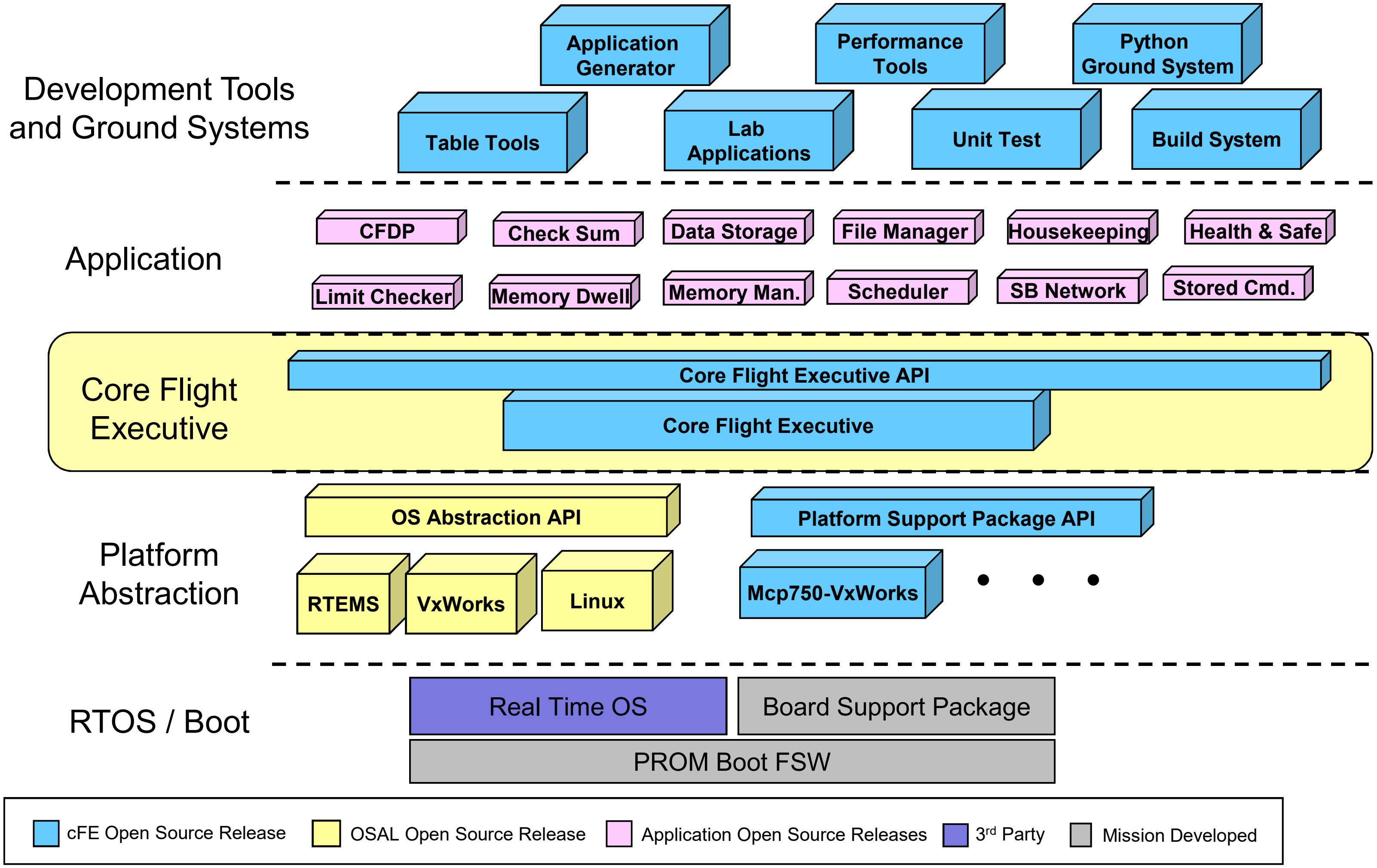
Core Flight System (cFS) Training

Module 2c: Event Services

Course Agenda

1. **Introduction**
2. **cFE Services**
   - a) Executive Services
   - b) Time Services
   - c) Event Services
   - d) Software Bus
   - e) Table Services
3. **Application Layer**
   - a) cFS Applications
   - b) cFS Libraries

Event Services - cFS Context



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

Board Support Package

PROM Boot FSW

cFE Open Source Release

OSAL Open Source Release

Application Open Source Releases

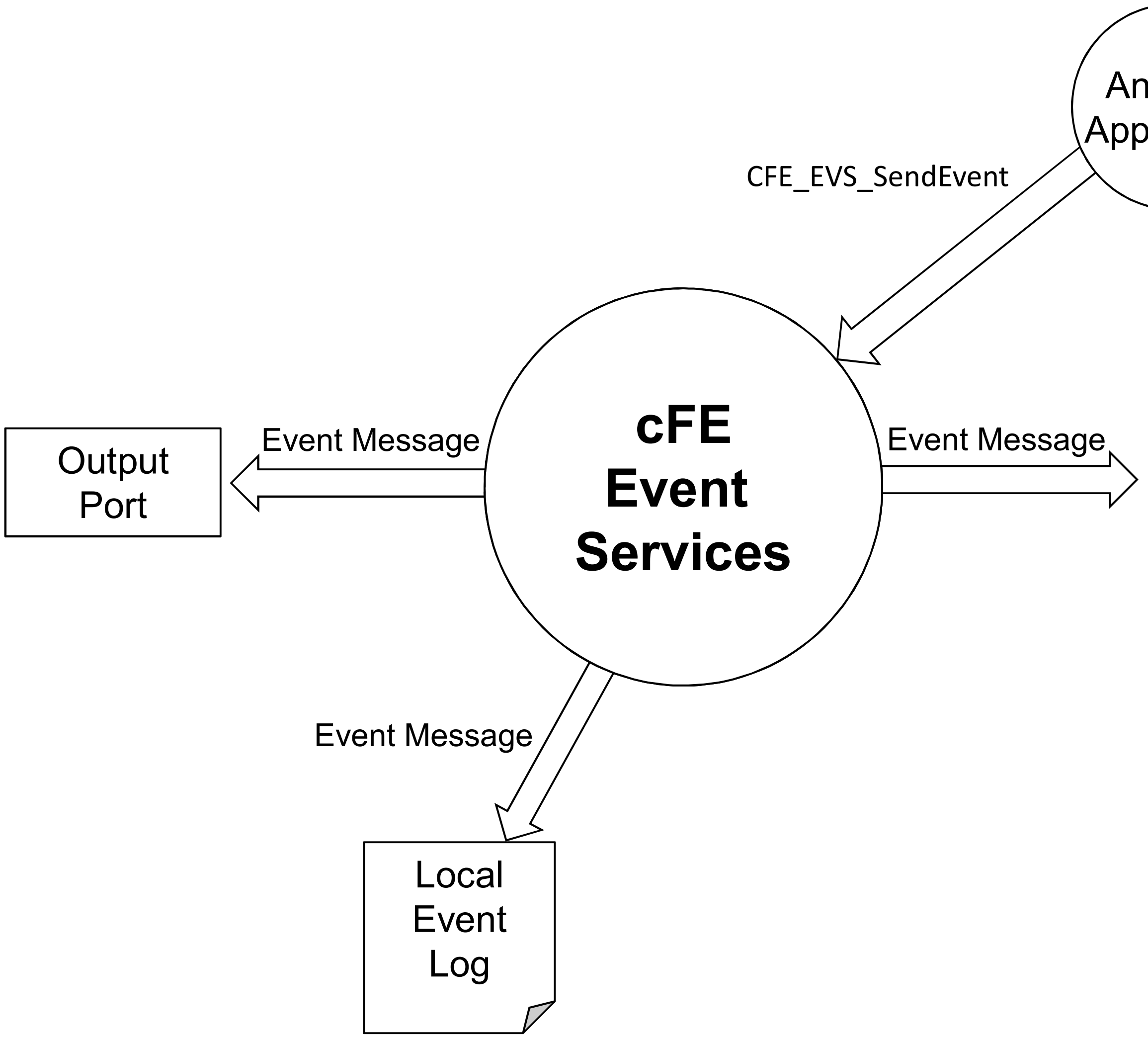
3rd Party

Mission Developed

Event Services (EVS) - Overview

- **Provides an interface for sending time-stamped text messages on the software bus**
  - Considered asynchronous because they are not part of telemetry periodically generated by an application
  - Processor unique identifier
  - Optionally logged to a local event log
  - Optionally output to a hardware port
- **Four event types defined**
  - Debug, Informational, Error, Critical
- **Event message control**
  - Apps can filter individual messages based on identifier
  - Enable/disable event types at the processor and application scope

Event Services - Context



Event Services – Message Format

- **Spacecraft time**
  - Retrieved via CFE_TIME_GetTime()

```
14:14:40.500 ERROR  CPU=CPU3  APPNAME=CFE_TBL  EVENT ID=57   Unable to locate “TST_TBL.invalid_tbl_02 in Table Registry
```

- **Event Type**
  - Debug, Informational, Error, Critical

```
14:14:40.500 ERROR CPU=CPU3  APPNAME=CFE_TBL  EVENT ID=57   Unable to locate “TST_TBL.invalid_tbl_02 in Table Registry
```

- **Spacecraft ID (not shown) defined in cfe_mission_cfg.h**
- **Processor ID defined in cfe_platform_cfg.h**

```
14:14:40.500  ERROR  CPU=CPU3  APPNAME=CFE_TBL  EVENT ID=57   Unable to locate “TST_TBL.invalid_tbl_02 in Table Registry
```

Event Services – Message Format

- **Application**
  - cFE Service or app name defined in cfe_es_startup.scr

```
14:14:40.500  ERROR  CPU=CPU3  APPNAME=CFE_TBL  EVENT ID=57   Unable to locate “TST_TBL.invalid_tbl_02 in Table Registry
```

- **Event ID is unique within an application**

```
14:14:40.500  ERROR  CPU=CPU3  APPNAME=CFE_TBL  EVENT ID=57  Unable to locate “TST_TBL.invalid_tbl_02 in Table Registry
```

- **Event Text is created using printf() format options**
  - “Short Format” platform option allows messages to be sent without text portion

```
14:14:40.500  ERROR  CPU=CPU3  APPNAME=CFE_TBL  EVENT ID=57  Unable to locate “TST_TBL.invalid_tbl_02 in Table Registry
```

Event Services – Event Filtering

- **Applications register events for filtering during initialization**
  - Registering immediately after ES app registration allows events to be used rather than syslog writes

- **Bit-wise AND "filter mask"**
  - Boolean AND performed on event ID message counter, if result is zero then the event is sent
  - Mask applied before the sent counter is incremented
  - 0x0000 => Every message sent
  - 0x0003 => Every 4th message sent
  - 0xFFFE => Only first two messages sent

- **CFE_EVS_MAX_FILTER_COUNT (cfe_evs_task.h) defines maximum count for a filtered event ID**
  - Once reached event becomes locked
  - Prevents erratic filtering behavior with counter rollover
  - Ground can unlock filter by resetting or deleting the filter

Event Services – No Filtering Example

Explicit Filter

```
static CFE_EVS_BinFilter_t  CFE_TO_EVS_Filters[] =
     {/* Event ID mask */
          {TO_INIT_INF_EID,          0x0000},
          {TO_CRCMDPIPE_ERR_EID,     0x0000},
          {TO_SUBSCRIBE_ERR_EID,     0x0000},
          {TO_TLMOUTSOCKET_ERR_EID,  0x0000},
          {TO_TLMOUTSTOP_ERR_EID,    0x0000},
          {TO_MSGID_ERR_EID,         0x0000},
          {TO_FNCODE_ERR_EID,        0x0000},
          {TO_NOOP_INF_EID,          0x0000}
     };

CFE_EVS_Register(CFE_TO_EVS_Filters,
               sizeof(CFE_TO_EVS_Filters)/sizeof(CFE_EVS_BinFilter_t),
               CFE_EVS_EventFilter_BINARY);
```

The “Explicit Filter” pattern is used for adding non-empty filters

NULL Filter

```
CFE_EVS_Register(NULL, 0, CFE_EVS_BINARY_FILTER);
```

or

```
CFE_EVS_Register(NULL, 0, CFE_EVS_NO_FILTER);
```

Event Services - Ports

- **cFE supports up to 4 ports**
  - Port behavior can be customized in cfe_evs_utils.c
  - By default, all ports call OS_printf
- **Event messages are sent to enabled ports in addition to the software bus**
- **By default, enabled ports are defined with the configuration parameter: CFE_PLATFORM_EVS_PORT_DEFAULT**
  - Enabled ports can be changed in runtime with the command CFE_EVS_EnablePortsCmd

Event Services – Message Control

- **Processor scope**
  - Enable/disable event messages based on type
    - Debug, Information, Error, Critical
- **Application scope**
  - Enable/disable all events
  - Enable/disable based on type
- **Event message scope**
  - During initialization apps can register events for filtering for up to CFE_PLATFORM_EVS_MAX_EVENT_FILTERS defined in cfe_platform_cfg.h
  - Filters can be modified by command

Event Services – Reset Behavior

- **Power-on Reset**
  - No data preserved
  - Application initialization routines register with the service
  - If configured local event log enabled

- **Processor Reset**
  - If configured with an event log, preserves
    - Messages
    - Mode: Discard or Overwrite
    - Log Full and Overflow status

Event Services – Retrieving Onboard State

- **Housekeeping Telemetry**
  - Log Enabled, Overflow, Full, Enabled
  - For each App: AppID, Events Sent Count, Enabled

- **Write application data to file. For each app**
  - Active flag – Are events enabled
  - Event Count
  - For each filtered event
    - Event ID
    - Filter Mask
    - Event Count – Number of times Event ID has been issued

- **Local event log**
  - If enabled, events are written to a local buffer
  - Log “mode” can be set to over write or discard
  - Serves as backup to onboard-recorder during initialization or error scenarios
  - Suitable for multi-processor architectures
  - Command to write log to file

Event Services - System Integration and App Development

- **System Integration**
  - DEBUG logging level should be disabled in flight
  - Telemetry Output should subscribe to and downlink event messages
- **App Development**
  - Any app can subscribe to event messages (like any other software bus message)
  - An app must register with event services before it can send any events
    - Apps should write to the ES system log if event services cannot be registered
  - Apps can send events with `CFE_EVS_SendEvent` or `CFE_EVS_SendTimedEvent`
    - These calls will have no effect if the app is not registered with EVS
  - cFE libraries cannot register with EVS

Event Services - Key Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_EVS_START_TASK_PRIORITY	EVS Task Priority
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE	EVS Task Stack Size
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS	Maximum Number of Event Filters per Application
CFE_PLATFORM_EVS_LOG_ON	Enable or Disable EVS Local Event Log
CFE_PLATFORM_EVS_DEFAULT_LOG_FILE	Default Event Log Filename
CFE_PLATFORM_EVS_LOG_MAX	Maximum Number of Events in EVS Local Event Log
CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE	Default EVS Application Data Filename
CFE_PLATFORM_EVS_PORT_DEFAULT	Default EVS Output Port State
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG	Default EVS Event Type Filter Mask
CFE_PLATFORM_EVS_DEFAULT_LOG_MODE	Default EVS Local Event Log Mode
CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE	Default EVS Message Format Mode

Event Services - APIs

Application Functions	Purpose
CFE_EVS_Register	Register the application with event services. All Applications must register with EVS
CFE_EVS_Unregister	Cleanup internal structures used by the event manager
CFE_EVS_SendEvent	Request to generate a software event. Event message will be generated based on filter settings
CFE_EVS_SendEventWithAppID	Generate a software event as though it came from the specified cFE Application
CFE_EVS_SendTimedEvent	Generate a software event with a specific time tag
CFE_EVS_ResetFilter	Resets the calling application’s event filter for a single event ID
CFE_EVS_ResetAllFilters	Resets all of the calling application’s event filters

Event Services – Command List

Command List	Purpose
CFE_EVS_NoopCmd	This function processes "no-op" commands received on the EVS command pipe
CFE_EVS_ClearLogCmd	This function processes "clear log" commands received on the EVS command pipe
CFE_EVS_ReportHousekeepingCmd	Request for housekeeping status telemetry packet
CFE_EVS_ResetCountersCmd	This function resets all the global counter variables that are part of the task telemetry
CFE_EVS_SetFilterCmd	This routine sets the filter mask for the given event_id in the calling task's filter array
CFE_EVS_EnablePortsCmd	This routine sets the command given ports to an enabled state
CFE_EVS_DisablePortsCmd	This routine sets the command given ports to a disabled state
CFE_EVS_EnableEventTypeCmd	This routine sets the given event types to an enabled state across all registered applications
CFE_EVS_DisableEventTypeCmd	This routine sets the given event types to a disabled state across all registered applications
CFE_EVS_SetEventFormatModeCmd	This routine sets the Event Format Mode
CFE_EVS_EnableAppEventTypeCmd	This routine sets the given event type for the given application identifier to an enabled state

Event Services – Command List

Command List	Purpose
CFE_EVS_DisableAppEventTypeCmd	This routine sets the given event type for the given application identifier to a disabled state
CFE_EVS_EnableAppEventsCmd	This routine enables application events for the given application identifier
CFE_EVS_DisableAppEventsCmd	This routine disables application events for the given application identifier
CFE_EVS_ResetAppCounterCmd	This routine sets the application event counter to zero for the given application identifier
CFE_EVS_ResetFilterCmd	This routine sets the application event filter counter to zero for the given application identifier and event identifier
CFE_EVS_ResetAllFiltersCmd	This routine sets all application event filter counters to zero for the given application identifier
CFE_EVS_AddEventFilterCmd	This routine adds the given event filter for the given application identifier and event identifier
CFE_EVS_DeleteEventFilterCmd	This routine deletes the event filter for the given application identifier and event identifier
CFE_EVS_WriteAppDataFileCmd	This routine writes all application data to a file for all applications that have registered with the EVS

Exercise 4 - Command cFE Event Service

Part 1 – Test a Debug Event Message

1. Ensure that cFE is running
2. Open a new terminal
3. Start the ground system executable (as in Exercise 2)
4. Enable Telemetry (as in Exercise 2)
5. Send an EVS No-Op command
    - Click the "Display Page" button beside "Event Services (CPU1)"
    - Click the "Send" button beside "Event Services No-Op"
6. Send a CI_LAB command to change the PDU size
    - In the main command window, click the "Display Page" button beside "Command Ingest LAB"
    - Click the "Send" button beside "CI_MODIFY_PDU_FILESIZE_CC"
    - Enter "0" for both parameters and click "Send"

Nothing shows up in the cFE window - that is expected

Exercise 4 - Command cFE Event Service

Part 2 – Enable and Show a Debug Message

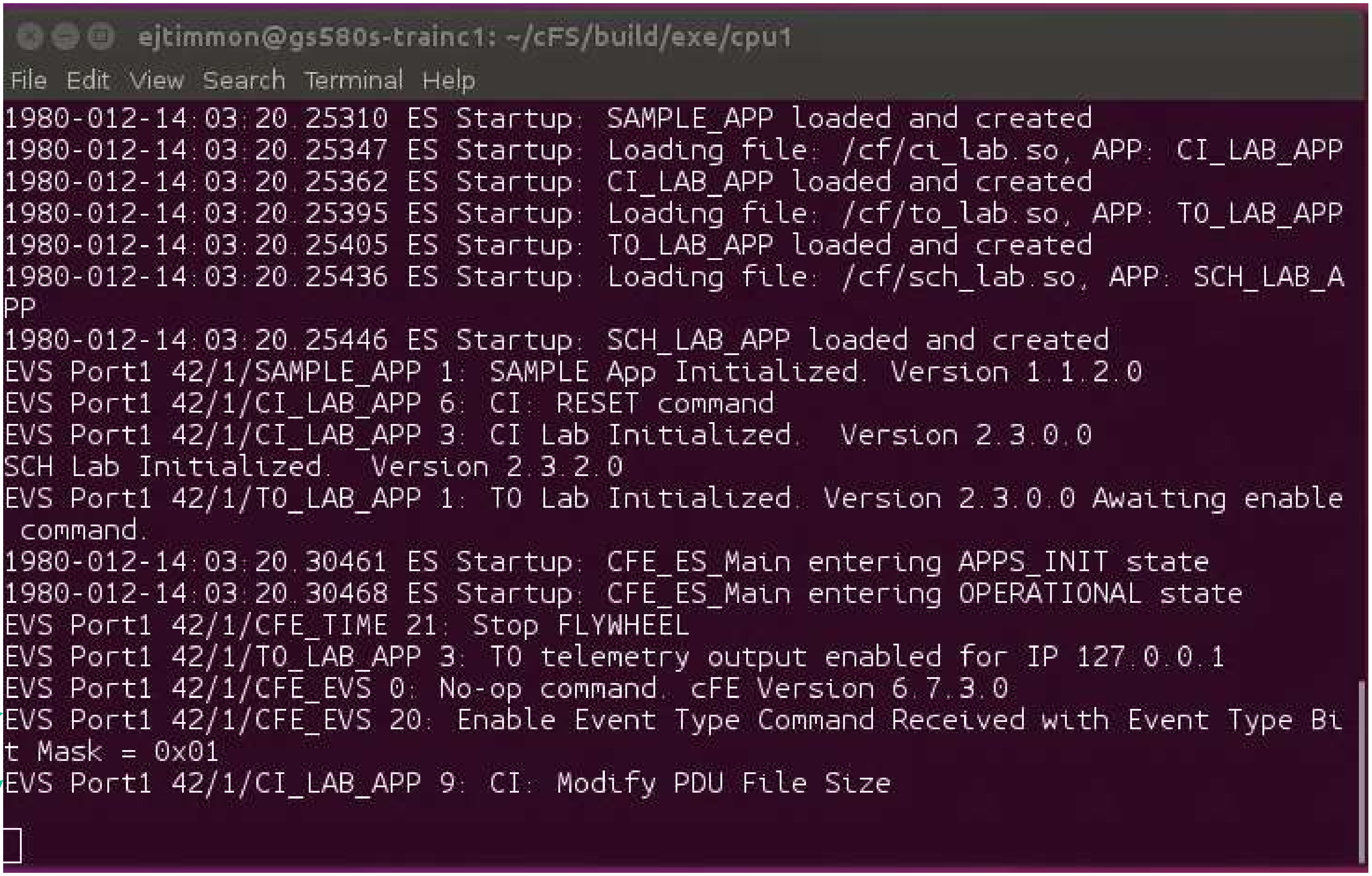
7. Send a command to enable debug messages
   - In the Event Services command window, click the "Send" button beside "Enable Event Type"
   - Enter "0x01" as the "BitMask" Input and "0x00" as the "Spare" input.
   - Click send

The "0x01" bitmask argument specifies the debug event type

8. Send a CI_LAB command to change the PDU size
   - In the main command window, click the "Display Page" button beside "Command Ingest LAB"
   - Click the "Send" button beside "CI_MODIFY_PDU_FILESIZE_CC"
   - Enter "0" for both parameters and click "Send"

Unlike the first time, a message should show up in the cFE window. This is because the CI_LAB event message associated with the PDU size command is a debug level event message. Therefore, it was disabled until command #7 enabled debug messages.

Exercise 4 Recap



ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1
File Edit View Search Terminal Help

```
1980-012-14:03:20.25310 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25347 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25362 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25395 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25405 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25436 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_A
PP
1980-012-14:03:20.25446 ES Startup: SCH_LAB_APP loaded and created
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
SCH Lab Initialized.  Version 2.3.2.0
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable
 command.
1980-012-14:03:20.30461 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30468 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 42/1/CFE_EVS 0: No-op command. cFE Version 6.7.3.0
EVS Port1 42/1/CFE_EVS 20: Enable Event Type Command Received with Event Type Bi
t Mask = 0x01
EVS Port1 42/1/CI_LAB_APP 9: CI: Modify PDU File Size
```

EVS No-Op Command

EVS Enable Event Type Command

CI_LAB Debug Message

National Aeronautics and Space Administration

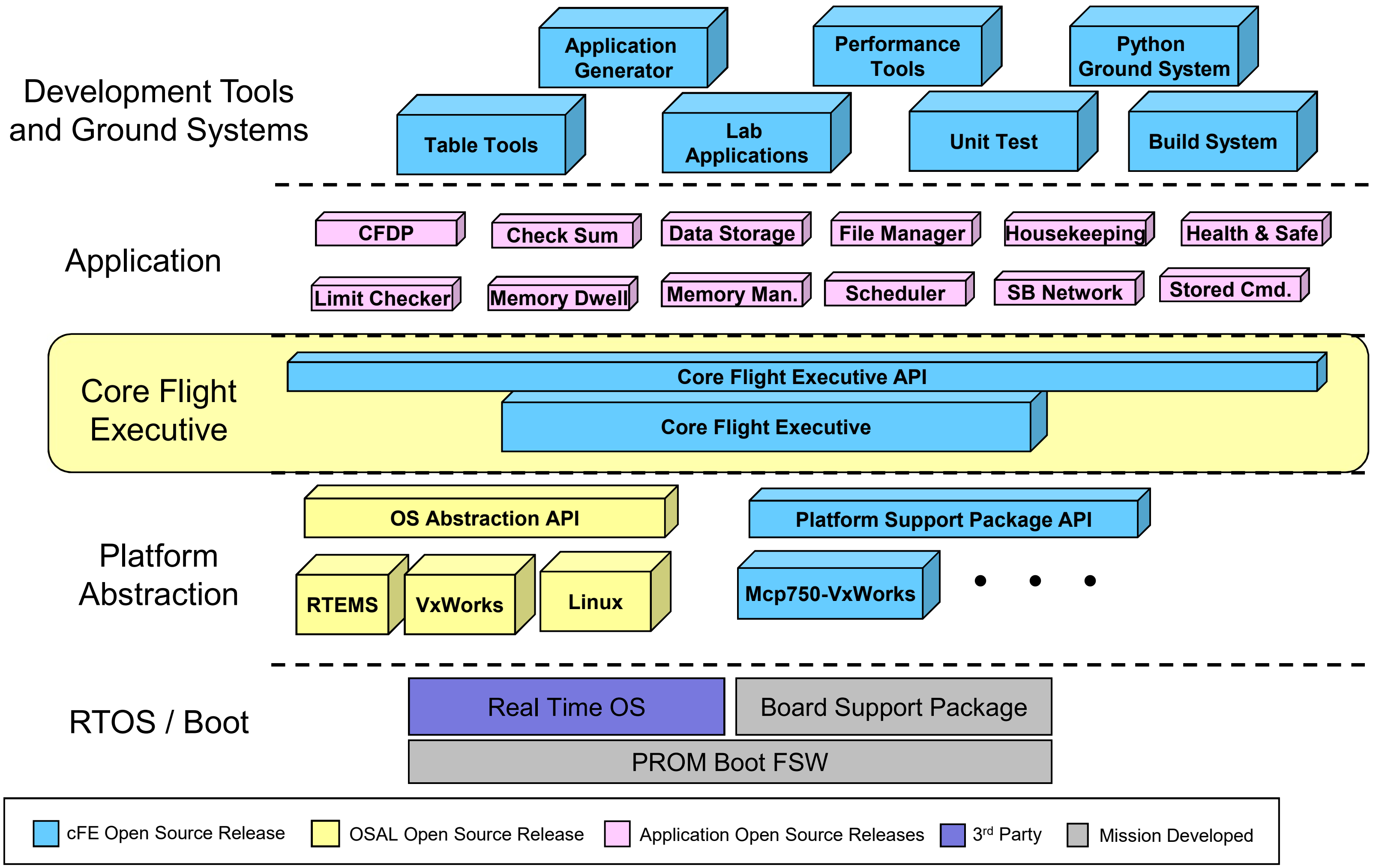
Core Flight System (cFS) Training

Module 2d: Software Bus Services

Course Agenda

1. **Introduction**
2. **cFE Services**
   a) Executive Services
   b) Time Services
   c) Event Services
   d) Software Bus
   e) Table Services
3. **Application Layer**
   a) cFS Applications
   b) cFS Libraries

Software Bus – cFS Context



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

Board Support Package

PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release □ Application Open Source Releases 🟪 3rd Party ⬜ Mission Developed

Software Bus (SB) Services - Overview

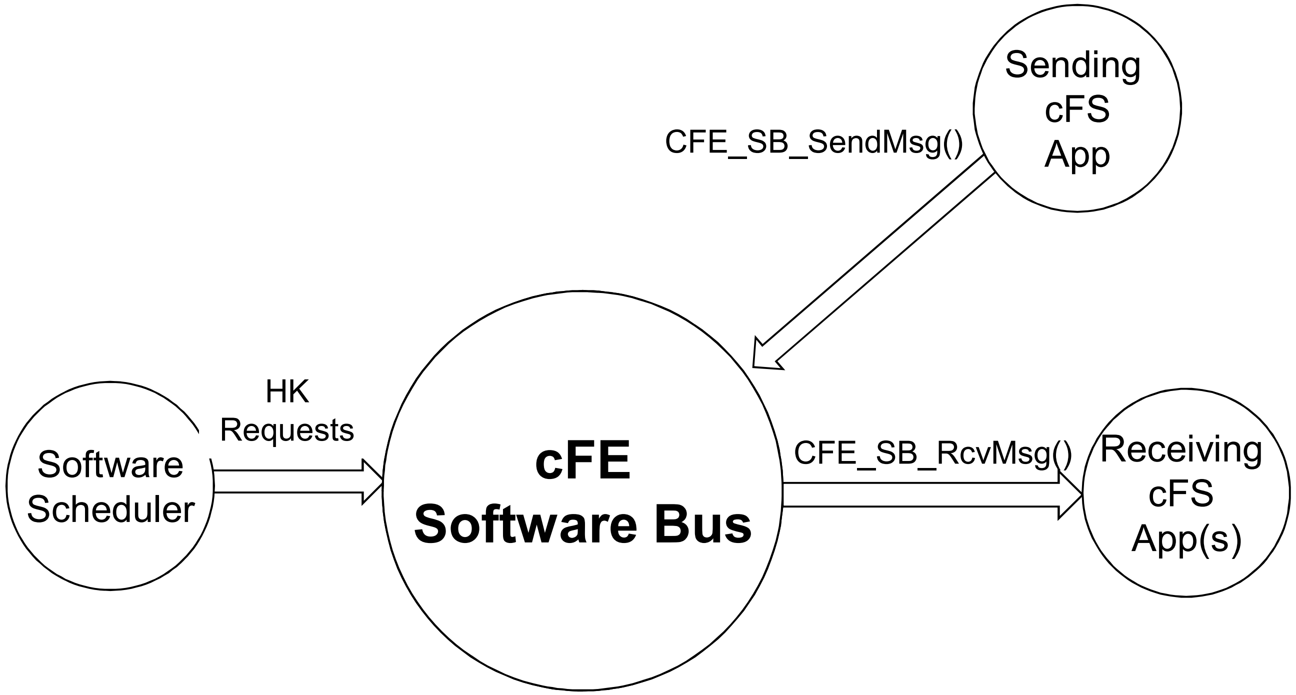
- **Provides a portable inter-application message service using a publish/subscribe model**

- **Routes messages to all applications that have subscribed to the message (i.e. broadcast model)**
  - Subscriptions are done at application startup
  - Message routing can be added/removed at runtime
  - Sender does not know who subscribes (i.e. connectionless)

- **Reports errors detected during the transferring of messages**

- **Outputs Statistics Packet and the Routing Information when commanded**

Software Bus - Context

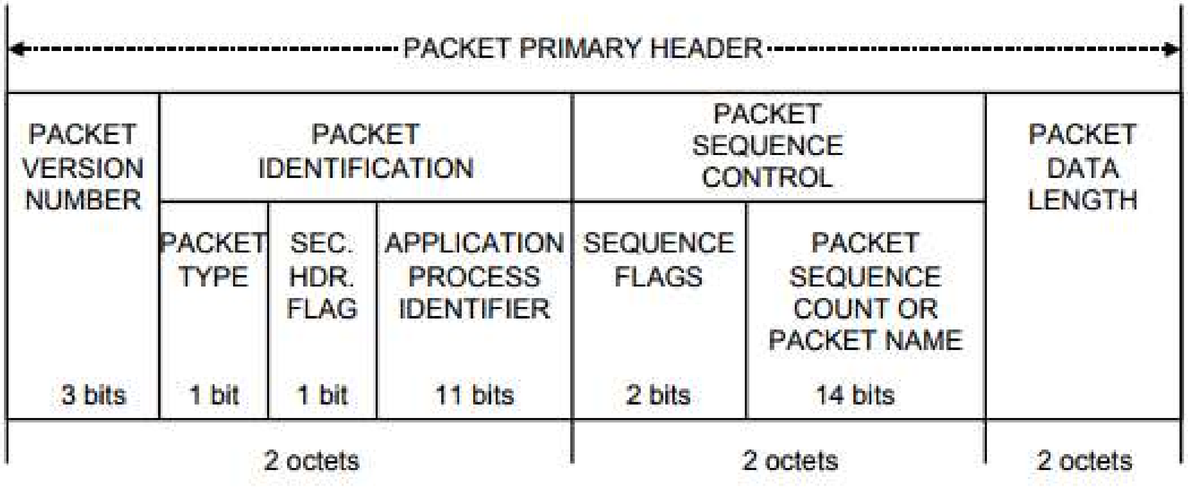


Software Bus – Messages (1 of 2)

- **Messages**
  - Data structures used to transfer data between applications

- **By default Consultative Committee for Space Data Systems (CCSDS) packets used to implement messages**
  - In theory other formats could be used but has not occurred in practice
  - Simplifies data management since CCSDS standards are used for flight-ground interfaces

- **CCSDS Primary Header (Always big endian)**



PACKET PRIMARY HEADER						
PACKET VERSION NUMBER	PACKET IDENTIFICATION			PACKET SEQUENCE CONTROL		PACKET DATA LENGTH
	PACKET TYPE	SEC. HDR. FLAG	APPLICATION PROCESS IDENTIFIER	SEQUENCE FLAGS	PACKET SEQUENCE COUNT OR PACKET NAME	
3 bits	1 bit	1 bit	11 bits	2 bits	14 bits	
2 octets				2 octets		2 octets

Software Bus – Messages (2 of 2)

- **“Packet” often used instead of “message” but not quite synonymous**
  - “Message ID” (first 16-bits) used to uniquely identify a message
  - “App ID” (11-bit) CCSDS packet identifier

- **Extended APID**
  - cFE 6.6 supports CCSDS extended APID, but testing has been limited

- **CCSDS Command Packets**
  - Secondary packet header contains a command function code
  - cFS apps typically define a single command packet and use the function code to dispatch a command processing function
  - Commands can originate from the ground or from onboard applications

- **CCSDS Telemetry Packets**
  - Secondary packet header contains a time stamp of when the data was produced
  - Telemetry is sent on the software bus by apps and can be ingested by other apps, stored onboard and sent to the ground

Software Bus – Message Formats

- **cFE abstracts the message format**
- **Implementation currently includes CCSDS format**
- **Software Bus provides functions to access message header (e.g. CFE_SB_SetCmdCode, CFE_SB_SetMsgTime etc.)**

```
typedef struct{
     CCSDS_PriHdr_t     Pri;
     CCSDS_CmdSecHdr_t  Sec;
} CFE_SB_CmdHdr_t;

typedef struct{
     CCSDS_PriHdr_t     Pri;
     CCSDS_TlmSecHdr_t  Sec;
} CFE_SB_TlmHdr_t;
```

Software Bus – Reset Behavior

- **No data is preserved for either a Power-On or Processor Reset**
  - All routing is reestablished as application create pipes and subscribe to messages
  - Any packet in transit at the time of the reset is discarded
  - All packet sequence counters reset to 1

Software Bus – Retrieving Onboard State

- **Telemetry**
  - Housekeeping Status
    - Counters (No subscribers, send errors, pipe overflows, etc.), Memory Stats

- **Telemetry packets generated by command**
  - Statistics
  - Subscription Report

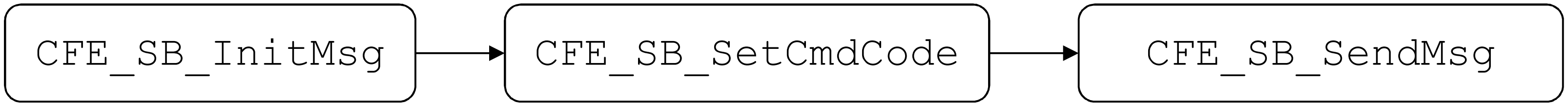
- **Files generated by command**
  - Routing Info
  - Pipe Info
  - Message ID to Route

Software Bus - System Integration

- **Message IDs should be unique across the system if possible**
- **The software bus places no restrictions on who can send or receive messages**
  - One-to-one
  - One-to-many
  - Many-to-one
  - Many-to-many
- **The Software Bus Network application can be used to extend the software bus across multiple processors**

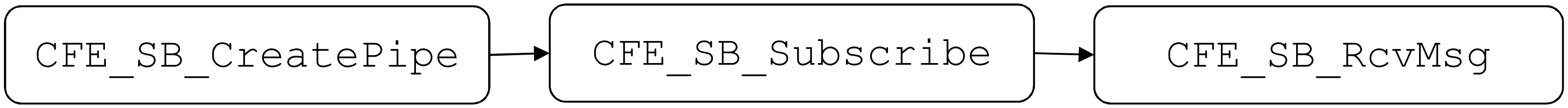
Software Bus – App Development (1 of 3)

- **Apps must create a pipe in order to receive messages**
  - Apps can create multiple pipes if necessary
- **Apps must subscribe to each individual message ID they want to receive**
  - Apps typically subscribe to at least 2 MIDs: one for housekeeping requests and one for commands
    - Commands are typically grouped under a single MID with multiple command codes
  - Apps can subscribe and unsubscribe to messages at any time
- **Sending Messages:**



CFE_SB_InitMsg → CFE_SB_SetCmdCode → CFE_SB_SendMsg

- **Receiving Messages:**



CFE_SB_CreatePipe → CFE_SB_Subscribe → CFE_SB_RcvMsg

Software Bus – App Development (2 of 3)

- **Multiple ways to send messages**

Function	Purpose
CFE_SB_SendMsg	Most basic and most common means of sending a message.
CFE_SB_PassMsg	Similar to CFE_SB_SendMsg, but intended for messages that are not generated by the sending application.
CFE_SB_ZeroCopySend	Eliminates an extra copy of the message. Can be used to improve performance. Requires the use of the helper function CFE_SB_ZeroCopyGetPtr
CFE_SB_ZeroCopyPass	

Software Bus – App Development (3 of 3)

- **Must first subscribe to messages**

Function	Purpose
CFE_SB_Subscribe	Subscribes to the message ID using default parameters for Quality of Service and Message Limit
CFE_SB_SubscribeEx	Subscribes to the message ID specifying custom parameters for Quality of Service and Message Limit

- **To receive messages, can pend or poll using the TimeOut parameter**

```
int32 CFE_SB_RcvMsg(CFE_SB_MsgPtr_t *BufPtr,
                    CFE_SB_PipeId_t  PipeId,
                    int32            TimeOut)
```

Software Bus – Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_SB_MAX_MSG_IDS	Maximum Number of Unique Message IDs SB Routing Table can hold
CFE_PLATFORM_SB_MAX_PIPES	Maximum Number of Unique Pipes SB Routing Table can hold
CFE_PLATFORM_SB_MAX_DEST_PER_PKT	Maximum Number of unique local destinations a single MsgId can have
CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT	Default Subscription Message Limit
CFE_PLATFORM_SB_BUF_MEMORY_BYTES	Size of the SB buffer memory pool
CFE_PLATFORM_SB_MAX_PIPE_DEPTH	Maximum depth allowed when creating an SB pipe
CFE_PLATFORM_SB_HIGHEST_VALID_MSGID	Highest Valid Message Id
CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME	Default Routing Information Filename
CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME	Default Pipe Information Filename
CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME	Default Message Map Filename
CFE_PLATFORM_SB_FILTERED_EVENT[1-8]	SB Event Filtering
CFE_PLATFORM_SB_FILTER_MASK[1-8]	SB Event Filtering Mask
CFE_PLATFORM_SB_MEM_BLOCK_SIZE_[01-16]	Define SB Memory Pool Block Sizes
CFE_PLATFORM_SB_MAX_BLOCK_SIZE	Defines Max SB Memory Pool Block Size
CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER	Default Sender Information Storage Mode
CFE_PLATFORM_SB_START_TASK_PRIORITY	SB Task Priority
CFE_PLATFORM_SB_START_TASK_STACK_SIZE	SB Task Stack Size

cFE Software Bus APIs

SB APIs	Purpose
CFE_SB_CreatePipe	API to create a pipe for receiving messages
CFE_SB_DeletePipe	Will unsubscribe to all routes associated with the given pipe id, then remove pipe from the pipe table
CFE_SB_SetPipeOpts	Sets pipe options
CFE_SB_GetPipeOpts	Gets the current pipe options
CFE_SB_SubscribeEx	API to globally subscribe to a message when QOS and MsgLim defaults are insufficient
CFE_SB_SubscribeLocal	CFE Internal API to locally subscribe to a message when QOS and MsgLim defaults are insufficient
CFE_SB_Subscribe	API to locally subscribe to a message when QOS and MsgLim defaults are sufficient
CFE_SB_Unsubscribe	API used to unsubscribe to a message
CFE_SB_UnsubscribeLocal	CFE Internal API used to locally unsubscribe to a message
CFE_SB_SendMsg	API used to send a message on the software bus
CFE_SB_PassMsg	API used to send a message on the software bus
CFE_SB_RcvMsg	API used to receive a message from the software bus
CFE_SB_GetLastSenderId	API used for receiving sender Information of the last message received on the given pipe
CFE_SB_ZeroCopyGetPtr	API used for getting a pointer to a buffer (for zero copy mode only)
CFE_SB_ZeroCopyReleasePtr	API used for releasing a pointer to a buffer (for zero copy mode only)
CFE_SB_ZeroCopySend	API for sending messages in zero copy mode (with telemetry source sequence count incrementing)
CFE_SB_ZeroCopyPass	API for sending messages in zero copy mode (telemetry source sequence count is preserved)

cFE Software Bus Utility APIs

SB Utility APIs	Purpose
CFE_SB_GetMsgId	Get the message ID of a software bus message
CFE_SB_SetMsgId	Set the message ID of a message in CCSDS header format
CFE_SB_MessageStringGet	Copies a string out of a software bus message
CFE_SB_MessageStringSet	Copies a string into a software bus message
CFE_SB_InitMsg	Initialize the header fields of a message
CFE_SB_MsgHdrSize	Get the size of a message header
CFE_SB_GetUserData	Get a pointer to the user data portion of a message
CFE_SB_GetUserDataLength	Get the length of the user data of a message (total size – header size)
CFE_SB_SetUserDataLength	Set the length field in the primary header
CFE_SB_GetTotalMsgLength	Get the total length of the message which includes the secondary header and the user data field
CFE_SB_SetTotalMsgLength	Set the length field, given the total length of the message
CFE_SB_GetMsgTime	Get the time field from a message
CFE_SB_SetMsgTime	Set the time field from a message
CFE_SB_TimeStampMsg	Set the time field to the current time
CFE_SB_GetCmdCode	Get the opcode field of message
CFE_SB_SetCmdCode	Set the opcode field of message
CFE_SB_GetChecksum	Get the checksum field of message
CFE_SB_GenerateChecksum	Calculate and Set the checksum field of message
CFE_SB_ValidateChecksum	Validate the checksum field of message

cFE Software Bus Command List

SB Command List	Purpose
CFE_SB_NoopCmd	Handler function the SB command
CFE_SB_ResetCountersCmd	Handler function the SB command
CFE_SB_EnableSubReportingCmd	Handler function the SB command
CFE_SB_DisableSubReportingCmd	Handler function the SB command
CFE_SB_SendHKTlmCmd	Function to send the SB housekeeping packet
CFE_SB_EnableRouteCmd	SB internal function to enable a specific route
CFE_SB_DisableRouteCmd	SB internal function to disable a specific route
CFE_SB_SendStatsCmd	SB internal function to send a Software Bus statistics packet
CFE_SB_SendRoutingInfoCmd	SB internal function to handle processing of 'Send Routing Info' command
CFE_SB_SendPipeInfoCmd	SB internal function to handle processing of 'Send Pipe Info' command
CFE_SB_SendMapInfoCmd	SB internal function to handle processing of 'Send Map Info' command
CFE_SB_SendPrevSubsCmd	SB function to build and send an SB packet containing a complete list of current subscriptions
CFE_SB_GetPipeName	Get the pipe name for a given ID
CFE_SB_GetPipeIdByName	Get the pipe ID by pipe name

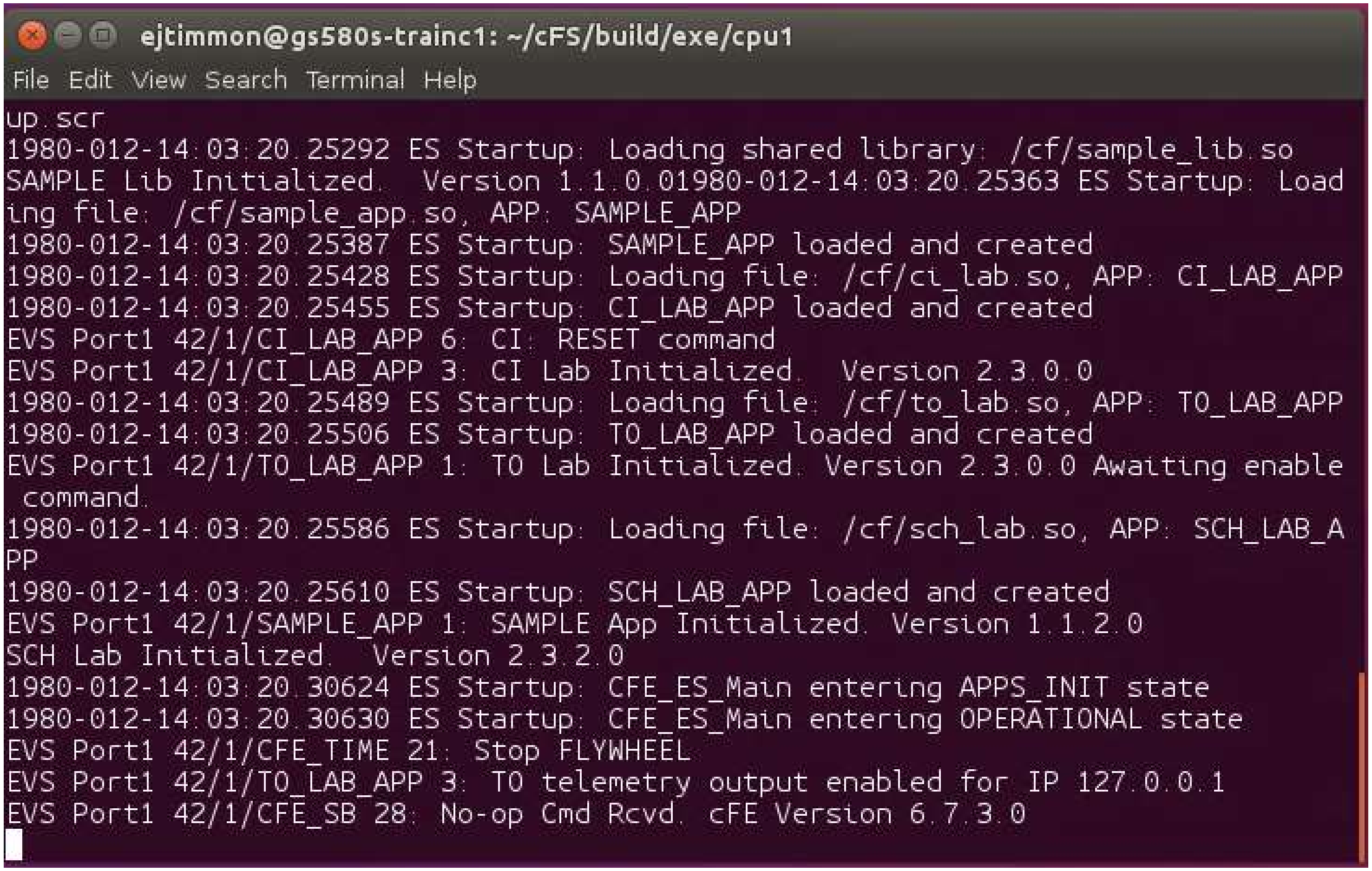
Exercise 5 - Command cFE Software Bus

1. Ensure that cFE is running
2. Open a new terminal
3. Start the ground system executable (as in Exercise 2)
4. Enable Telemetry (as in Exercise 2)
5. Send an SB No-Op command
   - Click the "Display Page" button beside "Software Bus (CPU1)"
   - Click the "Send" button beside "Software Bus No-Op"
   - Click "Send"
6. Send a "Write Map Info to a File" command
   - In the "Software Bus (CPU1)" window, click the "Send" button beside "Write Map Info to a File"
   - Enter "/cf/map.bin" in the "Input" field next to "Filename"
   - Click "Send"

Nothing appears in the cFE window unless debug messages have been enabled, but the file "map.bin" now exists in the build/exe/cpu1/cf directory. View with "hexdump -C cf/map.bin"

NOTE: The "Write Map Info to a File" command is one of several commands that together provide the full routing information for the software bus. This can be useful for troubleshooting purposes

Exercise 5 Recap


ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1
File Edit View Search Terminal Help

```
up.scr
1980-012-14:03:20.25292 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized.  Version 1.1.0.01980-012-14:03:20.25363 ES Startup: Load
ing file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25387 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25428 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25455 ES Startup: CI_LAB_APP loaded and created
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
1980-012-14:03:20.25489 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25506 ES Startup: TO_LAB_APP loaded and created
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable
 command.
1980-012-14:03:20.25586 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_A
PP
1980-012-14:03:20.25610 ES Startup: SCH_LAB_APP loaded and created
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
SCH Lab Initialized.  Version 2.3.2.0
1980-012-14:03:20.30624 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30630 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 42/1/CFE_SB 28: No-op Cmd Rcvd. cFE Version 6.7.3.0
```

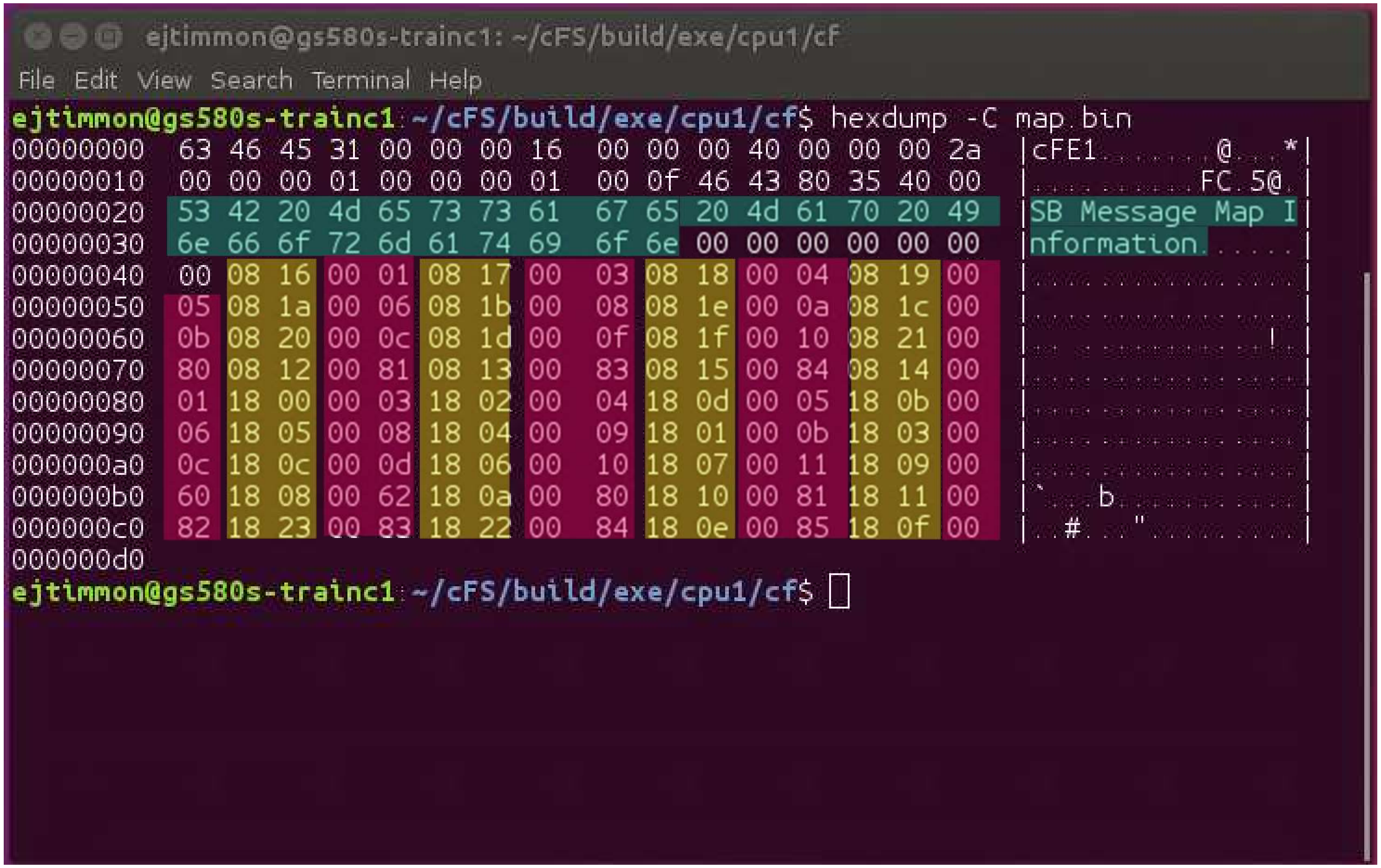
SB No-Op Command →

Exercise 5 Recap

🟩 File Header

🟨 Msg ID

🟥 Routing Table Index


ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1/cf
File Edit View Search Terminal Help

```
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1/cf$ hexdump -C map.bin
00000000  63 46 45 31 00 00 00 16  00 00 00 40 00 00 00 2a  |cFE1.......@...*|
00000010  00 00 00 01 00 00 00 01  00 0f 46 43 80 35 40 00  |..........FC.5@.|
00000020  53 42 20 4d 65 73 73 61  67 65 20 4d 61 70 20 49  |SB Message Map I|
00000030  6e 66 6f 72 6d 61 74 69  6f 6e 00 00 00 00 00 00  |nformation......|
00000040  00 08 16 00 01 08 17 00  03 08 18 00 04 08 19 00  |................|
00000050  05 08 1a 00 06 08 1b 00  08 08 1e 00 0a 08 1c 00  |................|
00000060  0b 08 20 00 0c 08 1d 00  0f 08 1f 00 10 08 21 00  |.. ...........!.|
00000070  80 08 12 00 81 08 13 00  83 08 15 00 84 08 14 00  |................|
00000080  01 18 00 00 03 18 02 00  04 18 0d 00 05 18 0b 00  |................|
00000090  06 18 05 00 08 18 04 00  09 18 01 00 0b 18 03 00  |................|
000000a0  0c 18 0c 00 0d 18 06 00  10 18 07 00 11 18 09 00  |................|
000000b0  60 18 08 00 62 18 0a 00  80 18 10 00 81 18 11 00  |`...b...........|
000000c0  82 18 23 00 83 18 22 00  84 18 0e 00 85 18 0f 00  |..#...".........|
000000d0
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1/cf$ 
```

CCSDS References

- Consultative Committee for Space Data Systems
- CCSDS Home: https://public.ccsds.org/default.aspx
- CCSDS Space Packet Protocol: https://public.ccsds.org/Pubs/133x0b1c2.pdf

National Aeronautics and Space Administration

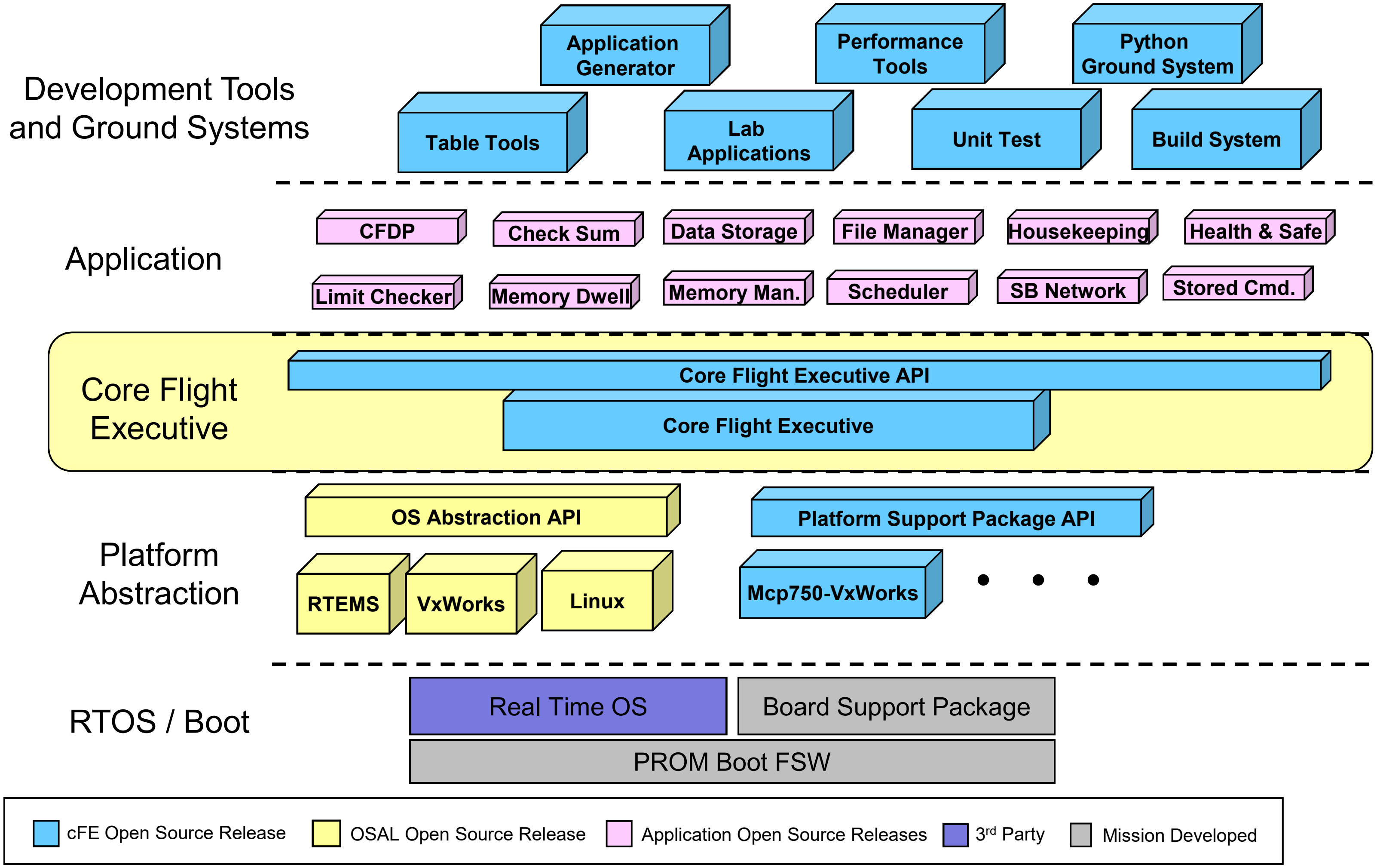
Core Flight System (cFS) Training

Module 2e: Table Services

Course Agenda

1. **Introduction**
2. **cFE Services**
   a) Executive Services
   b) Time Services
   c) Event Services
   d) Software Bus
   e) Table Services
3. **Application Layer**
   a) cFS Applications
   b) cFS Libraries

Table Services– cFS Context



Development Tools and Ground Systems

Application Generator
Performance Tools
Python Ground System
Table Tools
Lab Applications
Unit Test
Build System

Application

CFDP
Check Sum
Data Storage
File Manager
Housekeeping
Health & Safe
Limit Checker
Memory Dwell
Memory Man.
Scheduler
SB Network
Stored Cmd.

Core Flight Executive

Core Flight Executive API
Core Flight Executive

Platform Abstraction

OS Abstraction API
RTEMS
VxWorks
Linux
Platform Support Package API
Mcp750-VxWorks
• • •

RTOS / Boot

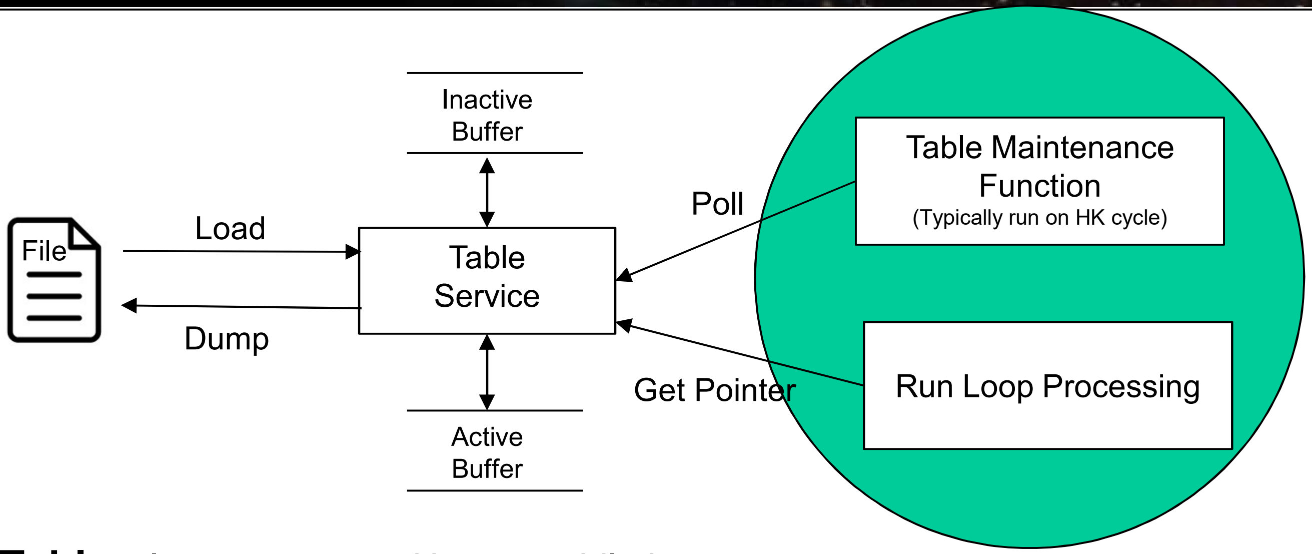
Real Time OS
Board Support Package
PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release 🟪 Application Open Source Releases 🟦 3rd Party ⬜ Mission Developed

Table Services (TBL) - Overview

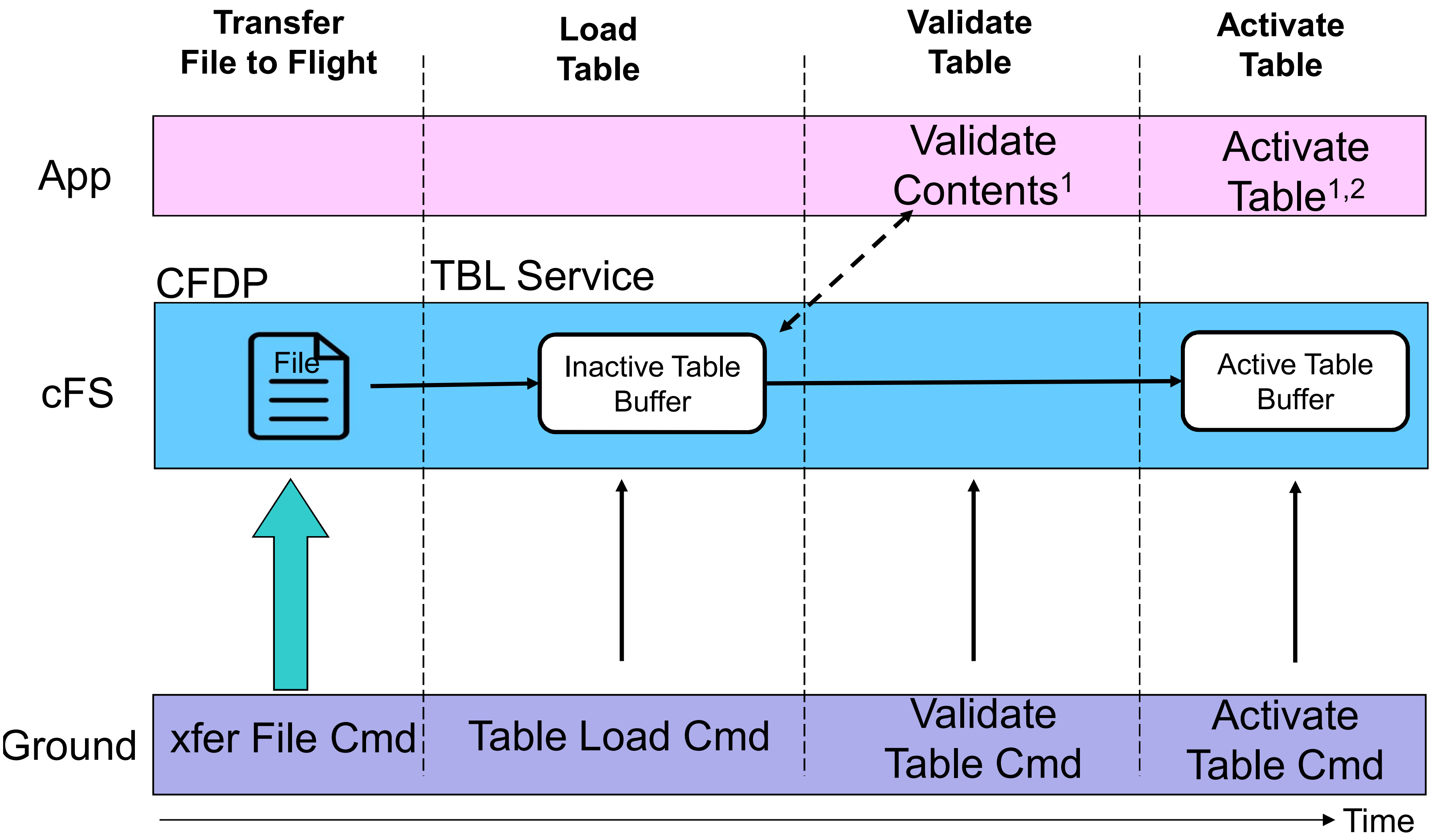
- **What is a table?**
  - Tables are logical groups of parameters that are managed as a named entity
- **Parameters typically change the behavior of a FSW algorithm**
  - Examples include controller gains, conversion factors, and filter algorithm parameters
- **Tables service provides ground commands to load a table from a file and dump a table to a file**
  - Table loads are synchronized with applications
- **Tables are binary files**
  - Ground support tools are required to create and display table contents
- **The cFE can be built without table support**
  - Note the cFE services don’t use tables

Table Services – Managing Tables



- **Active Table** - Image accessed by app while it executes
- **Inactive Table** - Image manipulated by ops (could be stored commands)
- **Load → Validate → Activate**
  - Loads can be partial or complete
  - For partial loads current active contents copied to inactive buffer prior to updates from file
  - Apps can supply a “validate function” that is executed when commanded
- **Dump**
  - Command specifies whether to dump the active or inactive buffer to a file
- **Table operations are synchronous with the application that owns the table to ensure table data integrity**
- **Non-Blocking table updates allow tables to be used in Interrupt Service Routines**

Table Services - Load Table



1. Apps typically validate & activate tables during their "housekeeping" execution cycle
2. In addition to instructing cFE to copy the contents, apps may have app-specific processing

Table Services - Dump Table

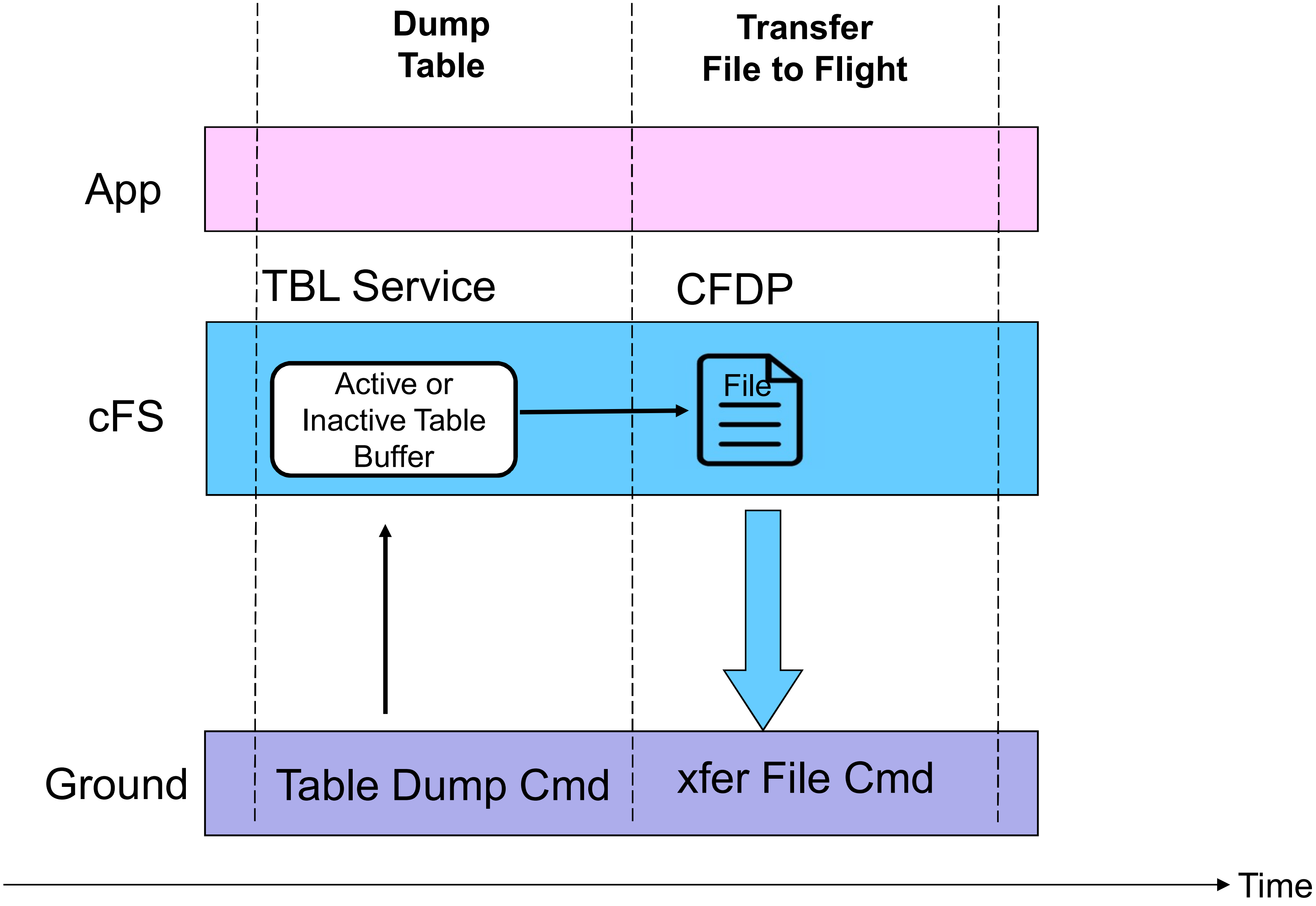


Table Services –Table Buffer Types

- **Single Buffer**
  - The active buffer is the only buffer dedicated to the application's table
  - Table service shares inactive buffers to service multiple app's with single buffer tables
    - CFE_TBL_MAX_SIMULTANEOUS_LOADS defines the number of concurrent table load sessions
  - Most efficient use of memory and adequate for most situations
  - Since

```
#define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)
```

- **Double Buffer**
  - Dedicated inactive image for each double buffered table
  - Useful for fast table image swaps (.e.g. high rate app and/or very large table) and delayed activation of table's content (e.g. ephemeris)
  - E.g. Stored Command's Absolute Time Command table
- **Shared single buffer pool must be sized to accommodate the largest single buffer image**

Table Services –Table Attributes

- **Validation Function**
  - Applications register validation functions during initialization
  - Table activates for tables with validation functions will be rejected if the validation has not been performed
  - Mission critical data table values are usually verified
- **Critical Tables**
  - Table data is stored in a Critical Data Store (CDS)
  - Contents updated for each table active command
- **User Defined Address**
  - Application provides the memory address for the active table buffer
  - Typically used in combination with a dump-only table
- **Dump-Only**
  - Contents can’t be changed via the load/validate/activate sequence
  - The dump is controlled by the application that owns the table so it can synchronize the dump and avoid dumps that contain partial updates

Table Services – Reset Behavior

- **Table registry is cleared for power-on and processor resets**
  - Applications must register tables for any type of reset
  - Applications must initialize their table data for any type of reset

- **Critical Table Exception**
  - If a table is registered as critical then during a processor reset table service will locate and load the preserved table data from a critical data store

Table Services – Retrieving Onboard State

- **Housekeeping Telemetry**
  - Table registry statistics (number of tables and pending loads)
  - Last table validation results (CRC, validation status, total validations)
  - Last updated table
  - Last file loaded
  - Last file umped
  - Last table loaded

- **Telemeter Application Registry**
  - Telemeter the Table Registry contents for the command-specified table

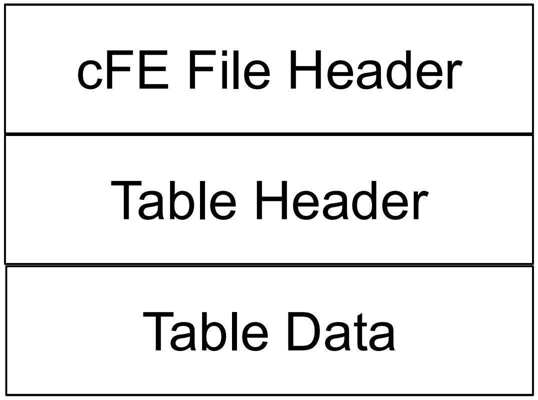
- **Dump Table Registry**
  - Write the pertinent table registry information to the command-specified file

Table Services
System Integration and App Development (1 of 2)

- **Commands are typically used to initiate an action; not tables**
  - For example, change a control mode
- **Sometimes convenience commands are provided to change table elements**
  - For example, scheduler app provides an enable/disable scheduler table entry
- **Typically tables do not contain dynamic data computed by the FSW**
  - The cFE doesn't preclude this and it has been used as a convenient method to collect data, save to a file, and transfer it to the ground
  - These are defined as dump-only tables
  - Static tables can be checksummed
- **Tables can be shared between applications but this is rare**
  - Tables are not intended to be an inter-application communication mechanism

Table Services
System Integration and App Development (2 of 2)

- **Load/dump files are binary files with the following sections:**



cFE File Header

Table Header

Table Data

- **Table header defined in cfe_tbl_internal.h**

```
{
   uint32   Reserved;   /**< Future Use: NumTblSegments in File?   */
   uint32   Offset;     /**< Byte Offset at which load should commence */
   uint32   NumBytes;   /**< Number of bytes to load into table */
   char     TableName[CFE_TBL_MAX_FULL_NAME_LEN]; /**< Fully qualified name of table */

} CFE_TBL_File_Hdr_t;
```

Table Services – Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_TBL_BUF_MEMORY_BYTES	Size of Table Services Table Memory Pool
CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE	Maximum Size Allowed for a Double Buffered Table
CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE	Maximum Size Allowed for a Single Buffered Table
CFE_PLATFORM_TBL_MAX_NUM_TABLES	Maximum Number of Tables Allowed to be Registered
CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES	Maximum Number of Critical Tables that can be Registered
CFE_PLATFORM_TBL_MAX_NUM_HANDLES	Maximum Number of Table Handles
CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS	Maximum Number of Simultaneous Loads to Support
CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS	Maximum Number of Simultaneous Table Validations
CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE	Default Filename for a Table Registry Dump
CFE_PLATFORM_TBL_VALID_SCID_COUNT	Number of Spacecraft ID's specified for validation
CFE_PLATFORM_TBL_U32FROM4CHARS	Macro to construct 32 bit value from 4 chars
CFE_PLATFORM_TBL_VALID_SCID_[1-2]	Spacecraft ID values used for table load validation
CFE_PLATFORM_TBL_VALID_PRID_COUNT	Number of Processor ID's specified for validation
CFE_PLATFORM_TBL_VALID_PRID_[1-4]	Processor ID values used for table load validation

Table Services APIs

Application Functions	Purpose
CFE_TBL_Register	Registers a new table
CFE_TBL_Unregister	Unregister a table and release its resources
CFE_TBL_Load	Initialize or update the contents of a table from memory or a file
CFE_TBL_Share	Get a handle to a table that was created by another application
CFE_TBL_GetAddress	Get the address of a table (locks the table)
CFE_TBL_GetAddresses	Get the address of a collection of tables (locks the tables)
CFE_TBL_ReleaseAddress	Release a table address (unlocks the table). Must be done periodically by the cFE Application that owns the table in order to allow updates to the tables
CFE_TBL_ReleaseAddresses	Release an array of table address (unlocks the tables)
CFE_TBL_GetStatus	Returns the status on the specified table regarding validation or update requests
CFE_TBL_Validate	Performs the registered validation function for the specified table and reports the success/failure to the operator via Table Services Housekeeping Telemetry and Event Messages.
CFE_TBL_Update	Update table contents with new data if an update is pending
CFE_TBL_Manage	Performs routine actions to manage the specified table. This includes performing any necessary table updates or table validations
CFE_TBL_GetInfo	Provides information about the specified table including size, last time updated etc.
CFE_TBL_DumpToBuffer	Copy Dump Only table to buffer for later dump to file by table services
CFE_TBL_Modified	Notify TBL Services that the contents of the table has been modified by the application
CFE_TBL_NotifyByMessage	Instruct TBL Services to notify calling application whenever the specified table requires management.

Table Services Commands

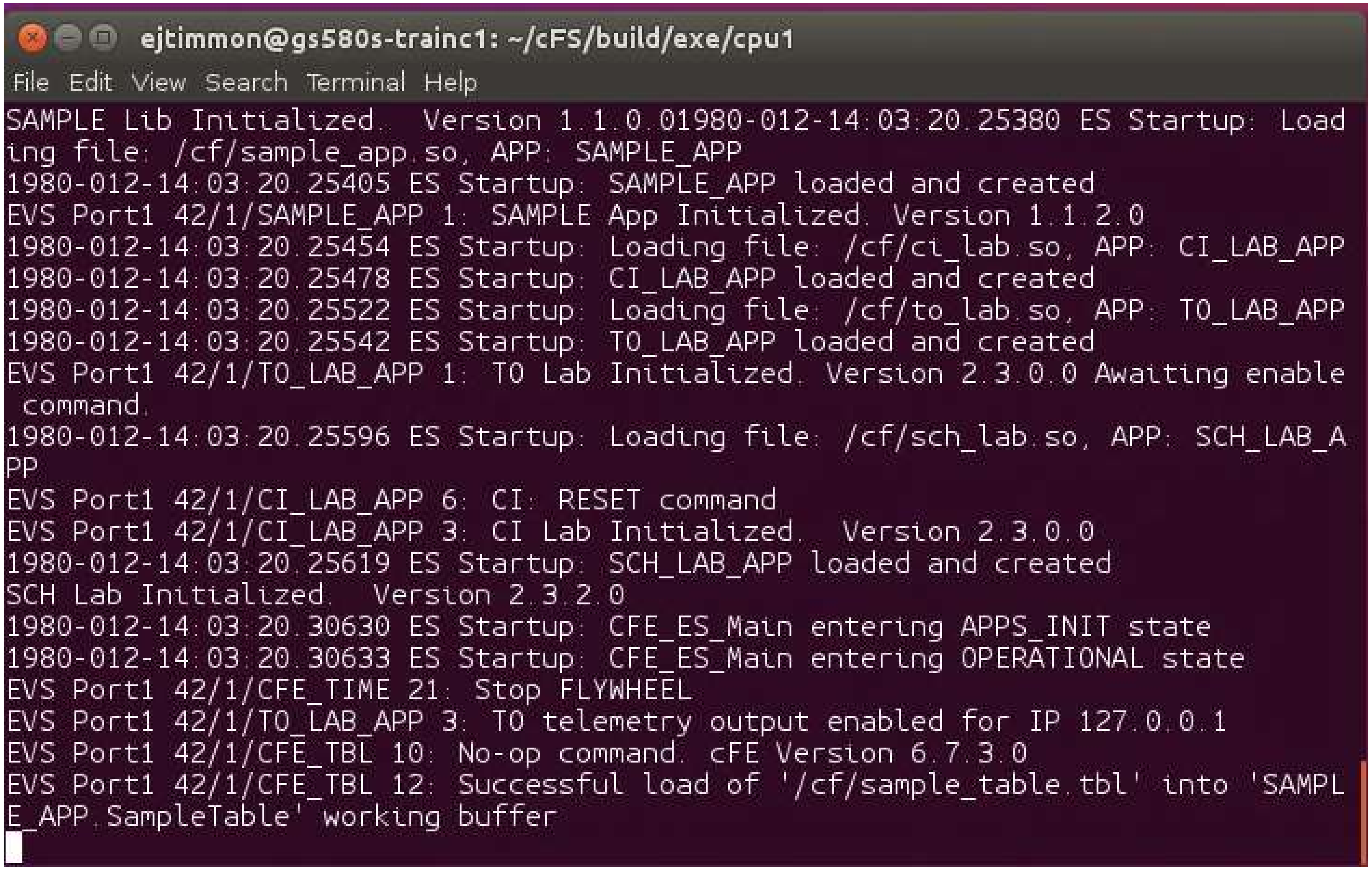
Command Functions	Purpose
CFE_TBL_HousekeepingCmd	Process Housekeeping Request Message
CFE_TBL_NoopCmd	Process NO-Op Command Message
CFE_TBL_ResetCountersCmd	Process Reset Counters Command Message
CFE_TBL_LoadCmd	Process Load Table File to Buffer Command Message
CFE_TBL_DumpCmd	Process Dump Table to File Command Message
CFE_TBL_ValidateCmd	Process Validate Table Command Message
CFE_TBL_ActivateCmd	Process Activate Table Command Message
CFE_TBL_DumpRegistryCmd	Process Dump Table Registry to file Command Message
CFE_TBL_SendRegistryCmd	Process Telemeter Table Registry Entry Command Message
CFE_TBL_DeleteCDSCmd	Process Delete Critical Table's CDS Command Message
CFE_TBL_AbortLoadCmd	Process Abort Load Command Message

Exercise 6 - Command cFE Table Service

1. Ensure that cFE is running
2. Open a new terminal
3. Start the ground system executable (as in Exercise 2)
4. Enable Telemetry (as in Exercise 2)
5. Send an TBL No-Op command
   - Click the "Display Page" button beside "Table Services (CPU1)"
   - Click the "Send" button beside "Table No-Op"
6. Send a "Load Table" command
   - In the "Table Services (CPU1)" window, click the "Send" button beside "Load Table"
   - Enter "/cf/sample_table.tbl" in the "Input" field next to "LoadFilename"
   - Click "Send"
7. Dump the table registry
   - In the "Table Services (CPU1)" window, click the "Send" button beside "Dump Table Registry"
   - Enter "/cf/tbl_reg.bin" in the "Input" field next to "DumpFilename"
   - Click "Send"

Nothing appears in the cFE window unless debug messages have been enabled, but the file "tbl_reg.bin" now exists in the build/exe/cpu1/cf directory. View with "hexdump -C cf/tbl_reg.bin"

Exercise 6 Recap

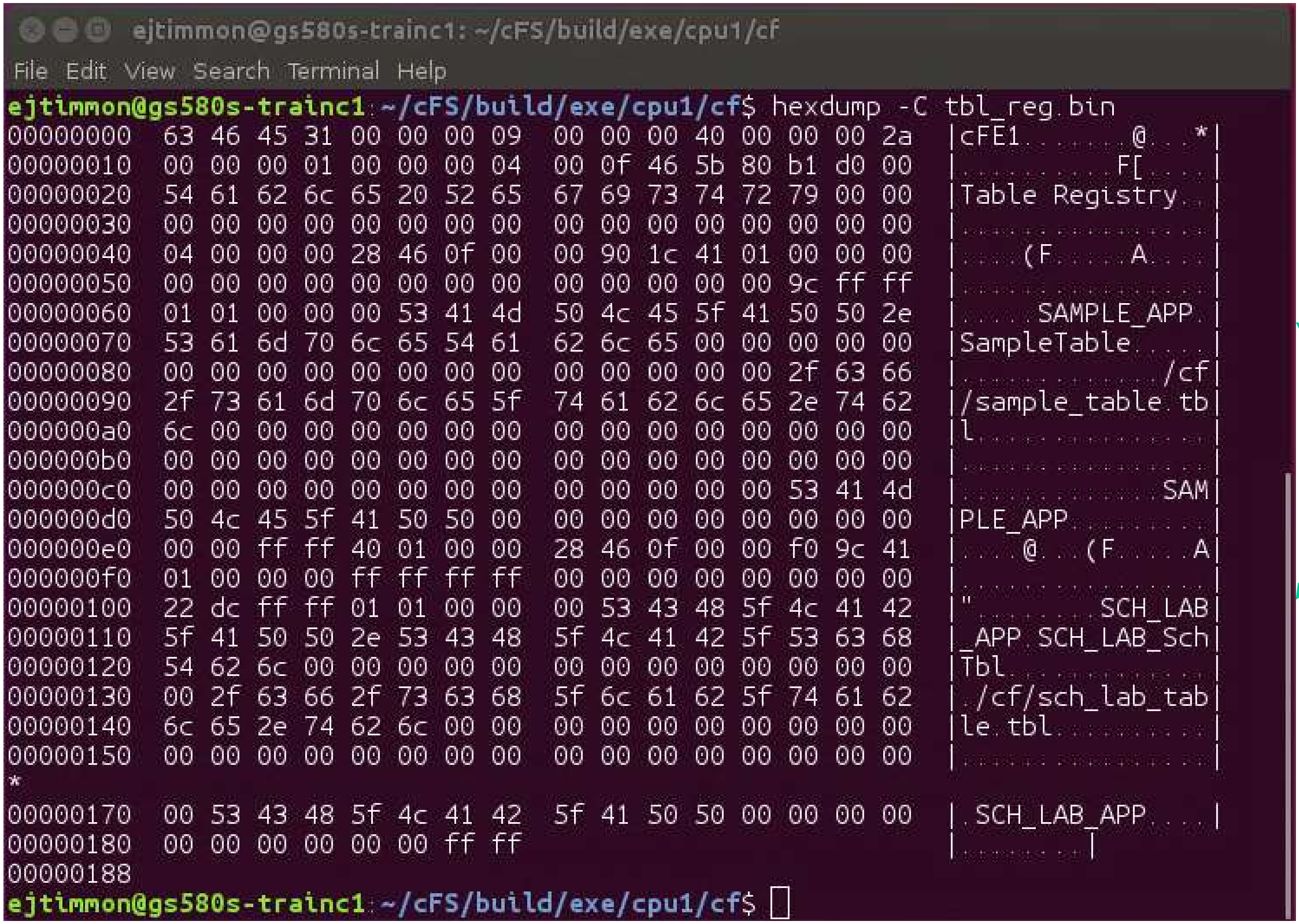

ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1
File Edit View Search Terminal Help

```
SAMPLE Lib Initialized.  Version 1.1.0.01980-012-14:03:20.25380 ES Startup: Load
ing file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25405 ES Startup: SAMPLE_APP loaded and created
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
1980-012-14:03:20.25454 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25478 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25522 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25542 ES Startup: TO_LAB_APP loaded and created
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable
 command.
1980-012-14:03:20.25596 ES Startup: Loading file: /cf/sch_lab.so, APP: SCH_LAB_A
PP
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
1980-012-14:03:20.25619 ES Startup: SCH_LAB_APP loaded and created
SCH Lab Initialized.  Version 2.3.2.0
1980-012-14:03:20.30630 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30633 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 42/1/CFE_TBL 10: No-op command. cFE Version 6.7.3.0
EVS Port1 42/1/CFE_TBL 12: Successful load of '/cf/sample_table.tbl' into 'SAMPL
E_APP.SampleTable' working buffer
```

TBL No-Op Command

TBL Load Command

Exercise 6 Recap



ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1/cf
File Edit View Search Terminal Help

```
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1/cf$ hexdump -C tbl_reg.bin
00000000  63 46 45 31 00 00 00 09  00 00 00 40 00 00 00 2a  |cFE1.......@...*|
00000010  00 00 00 01 00 00 00 04  00 0f 46 5b 80 b1 d0 00  |..........F[....|
00000020  54 61 62 6c 65 20 52 65  67 69 73 74 72 79 00 00  |Table Registry..|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  04 00 00 00 28 46 0f 00  00 90 1c 41 01 00 00 00  |....(F.....A....|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 9c ff ff  |................|
00000060  01 01 00 00 00 53 41 4d  50 4c 45 5f 41 50 50 2e  |.....SAMPLE_APP.|
00000070  53 61 6d 70 6c 65 54 61  62 6c 65 00 00 00 00 00  |SampleTable.....|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 2f 63 66  |............./cf|
00000090  2f 73 61 6d 70 6c 65 5f  74 61 62 6c 65 2e 74 62  |/sample_table.tb|
000000a0  6c 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |l...............|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 53 41 4d  |.............SAM|
000000d0  50 4c 45 5f 41 50 50 00  00 00 00 00 00 00 00 00  |PLE_APP.........|
000000e0  00 00 ff ff 40 01 00 00  28 46 0f 00 00 f0 9c 41  |....@...(F.....A|
000000f0  01 00 00 00 ff ff ff ff  00 00 00 00 00 00 00 00  |................|
00000100  22 dc ff ff 01 01 00 00  00 53 43 48 5f 4c 41 42  |"........SCH_LAB|
00000110  5f 41 50 50 2e 53 43 48  5f 4c 41 42 5f 53 63 68  |_APP.SCH_LAB_Sch|
00000120  54 62 6c 00 00 00 00 00  00 00 00 00 00 00 00 00  |Tbl.............|
00000130  00 2f 63 66 2f 73 63 68  5f 6c 61 62 5f 74 61 62  |./cf/sch_lab_tab|
00000140  6c 65 2e 74 62 6c 00 00  00 00 00 00 00 00 00 00  |le.tbl..........|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000170  00 53 43 48 5f 4c 41 42  5f 41 50 50 00 00 00 00  |.SCH_LAB_APP....|
00000180  00 00 00 00 00 00 ff ff                           |........|
00000188
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1/cf$
```

2 Tables in System

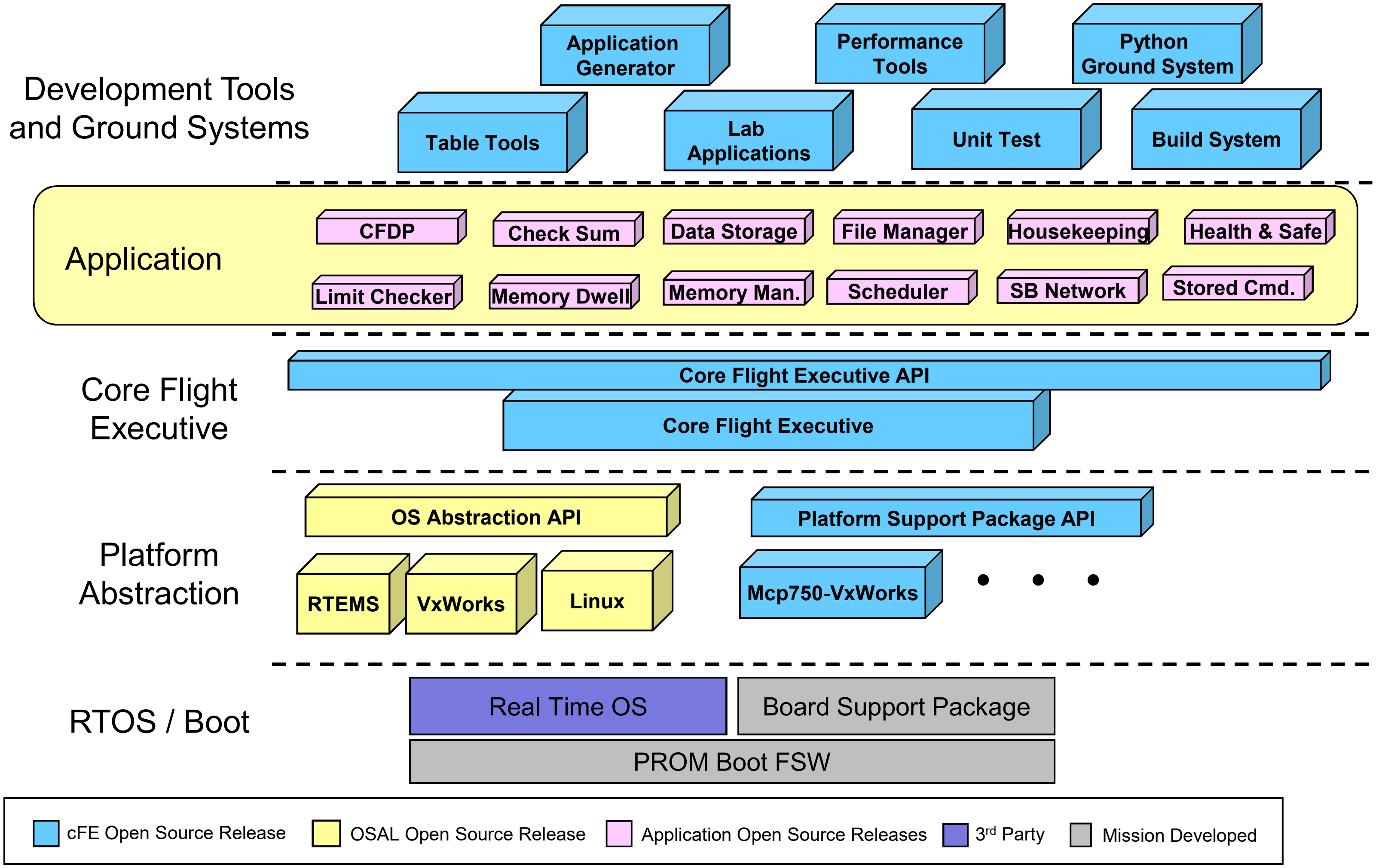
Core Flight Executive (cFS) Training

Module 3: Application Development

Course Agenda

1. **Introduction**
2. **cFE Services**
    a) Executive Services
    b) Time Services
    c) Event Services
    d) Software Bus
    e) Table Services
3. **Application Layer**
    a) cFS Applications
    b) cFS Libraries

cFE Services - cFS Context



Development Tools and Ground Systems

Application Generator

Performance Tools

Python Ground System

Table Tools

Lab Applications

Unit Test

Build System

Application

CFDP

Check Sum

Data Storage

File Manager

Housekeeping

Health & Safe

Limit Checker

Memory Dwell

Memory Man.

Scheduler

SB Network

Stored Cmd.

Core Flight Executive

Core Flight Executive API

Core Flight Executive

Platform Abstraction

OS Abstraction API

Platform Support Package API

RTEMS

VxWorks

Linux

Mcp750-VxWorks

• • •

RTOS / Boot

Real Time OS

Board Support Package

PROM Boot FSW

🟦 cFE Open Source Release 🟨 OSAL Open Source Release 🟪 Application Open Source Releases 🟦 3rd Party ⬜ Mission Developed

cFS Applications

- **Can run anywhere the cFS framework has been deployed**

- **Provide “higher level” functions than the cFE itself**
  - Command and data handling
  - Guidance, navigation, and control
  - Onboard data processing

- **GSFC has released 12 applications that provide common command and data handling functionality such as**
  - Stored command management and execution
  - Onboard data storage file management

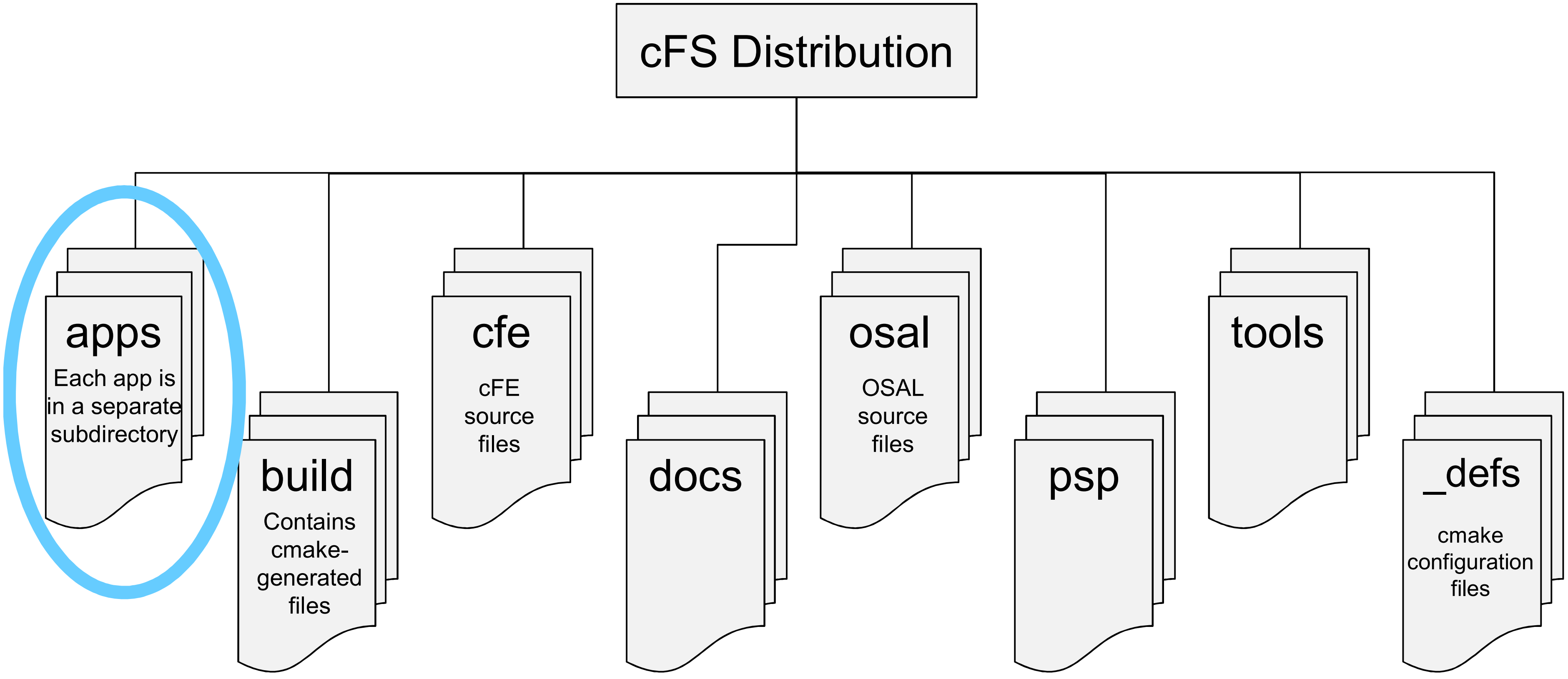
- **Missions use a combination of custom and reused applications**

cFS Libraries

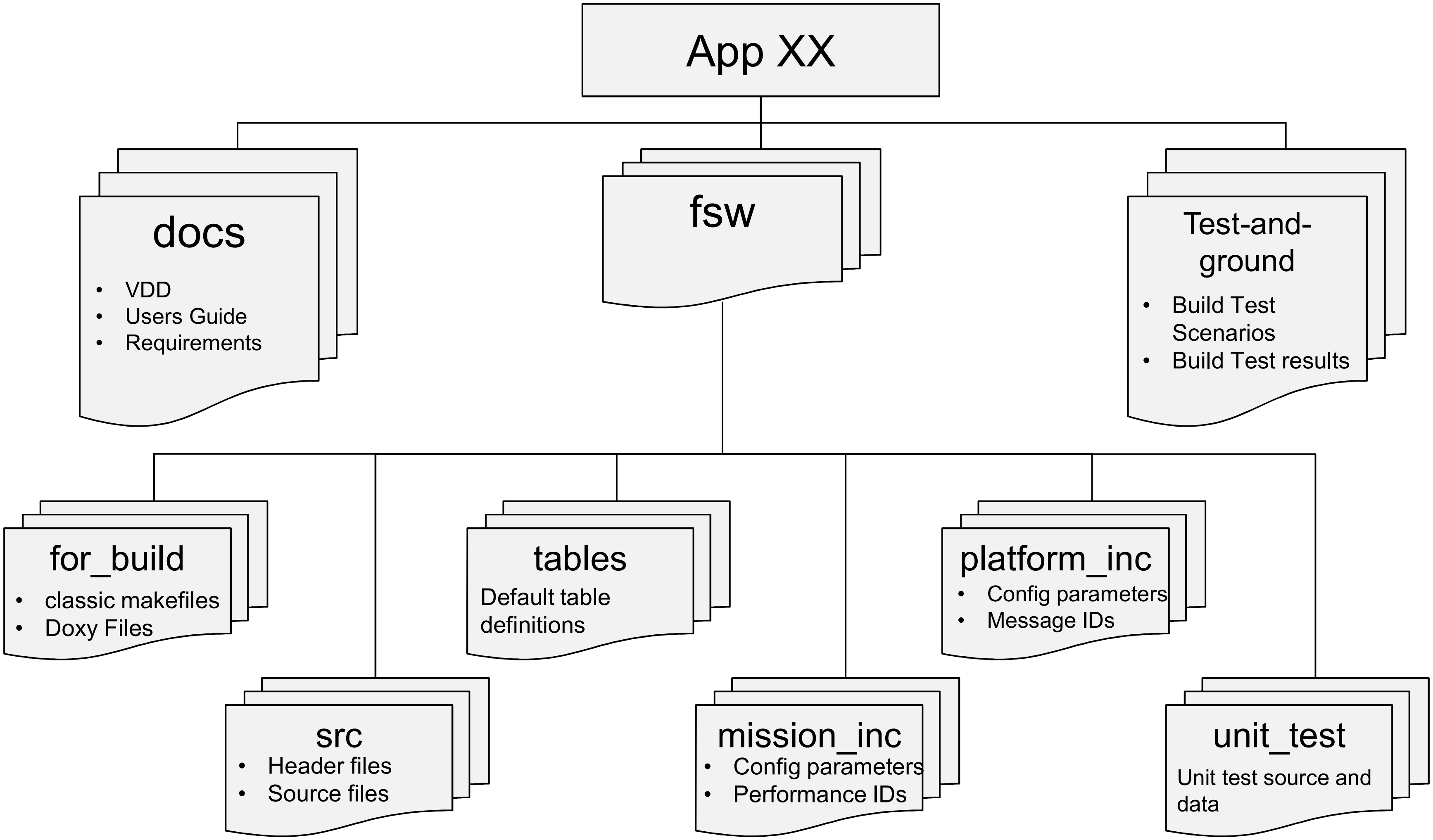
- **What is a library?**
  - A collection of utilities available for use by apps
  - No main task execution in the library
  - Exist at the application layer of the cFS
- **Specified in the cfe_es_startup.scr script and loaded at cFE startup**
- **Libraries can’t use application services that require registration**
  - e.g. Event Services
- **Checksum can’t do library code space**

Application Build Context

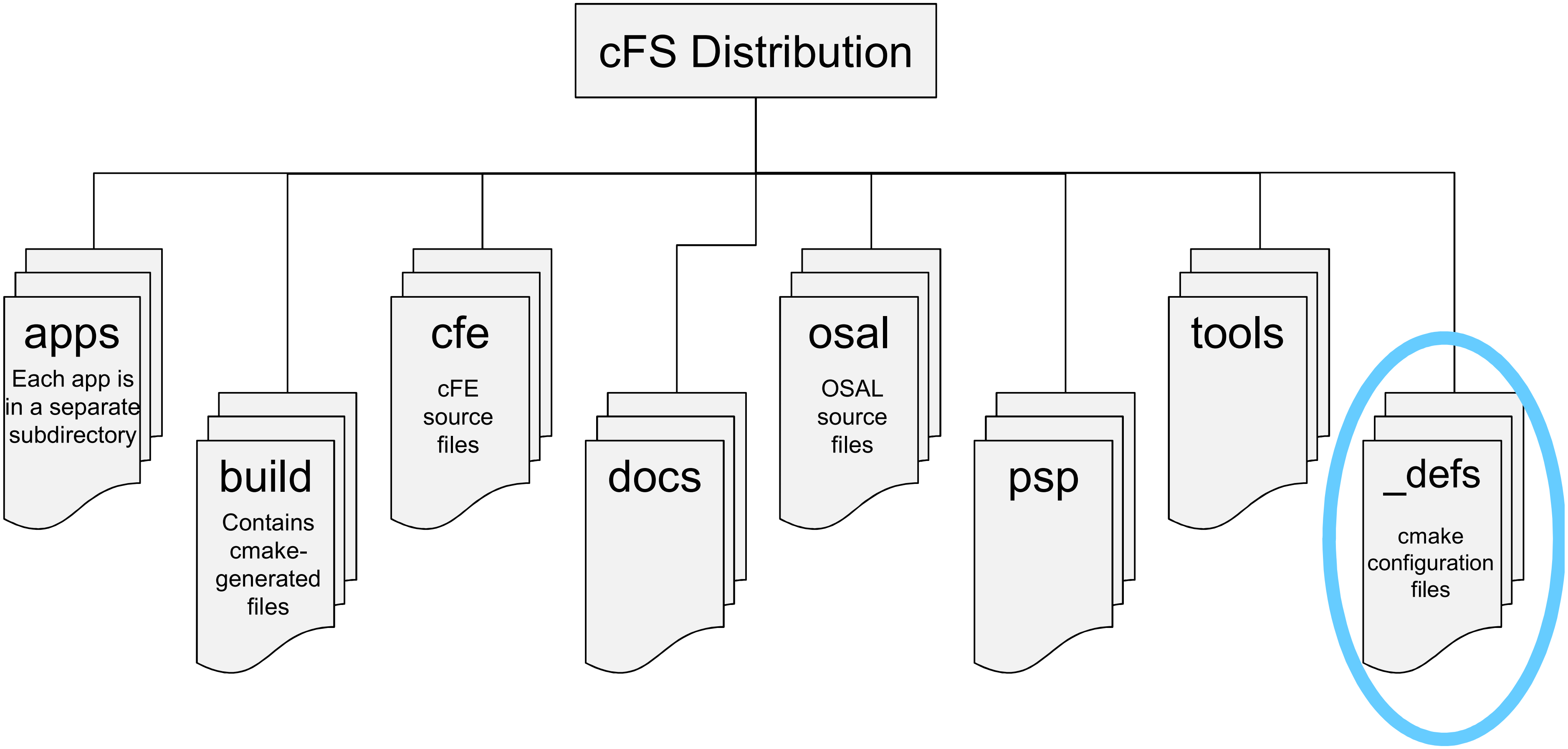
cFS Mission Directory Structure


cFS Distribution
apps
Each app is in a separate subdirectory
build
Contains cmake-generated files
cfe
cFE source files
docs
osal
OSAL source files
psp
tools
_defs
cmake configuration files

App Directory Structure



cFS Mission Directory Structure



cFS Distribution

apps

Each app is in a separate subdirectory

build

Contains cmake-generated files

cfe

cFE source files

docs

osal

OSAL source files

psp

tools

_defs

cmake configuration files

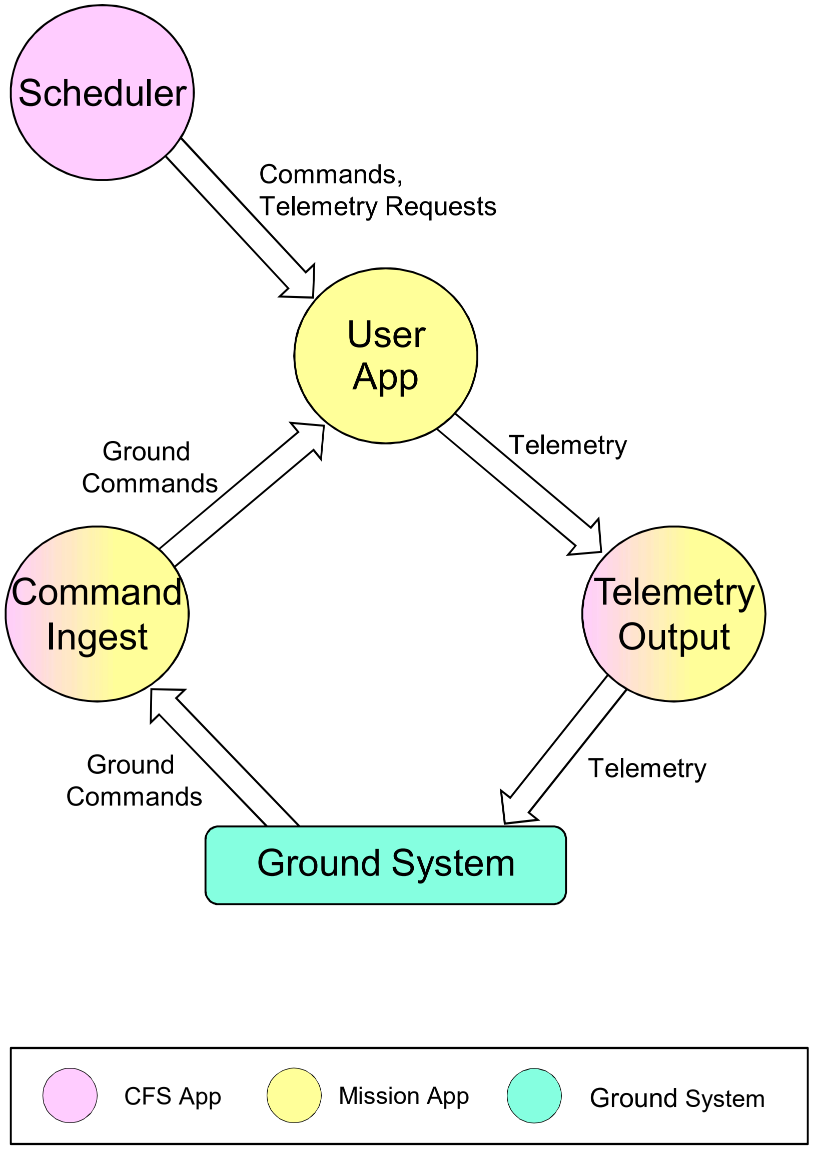
_def Directory Structure

- **Targets.cmake**
  - Identifies the target architectures and configurations
  - Identifies the apps to be built
  - Identifies files that will be copied from *_def to platform specific directories

- **Copied file examples**
  - cpu1_cfe_es_startup.scr
  - cpu1_msgids.h
  - cpu1_osconfig.h

Application Runtime Context

Application Runtime Context



Application Runtime Context

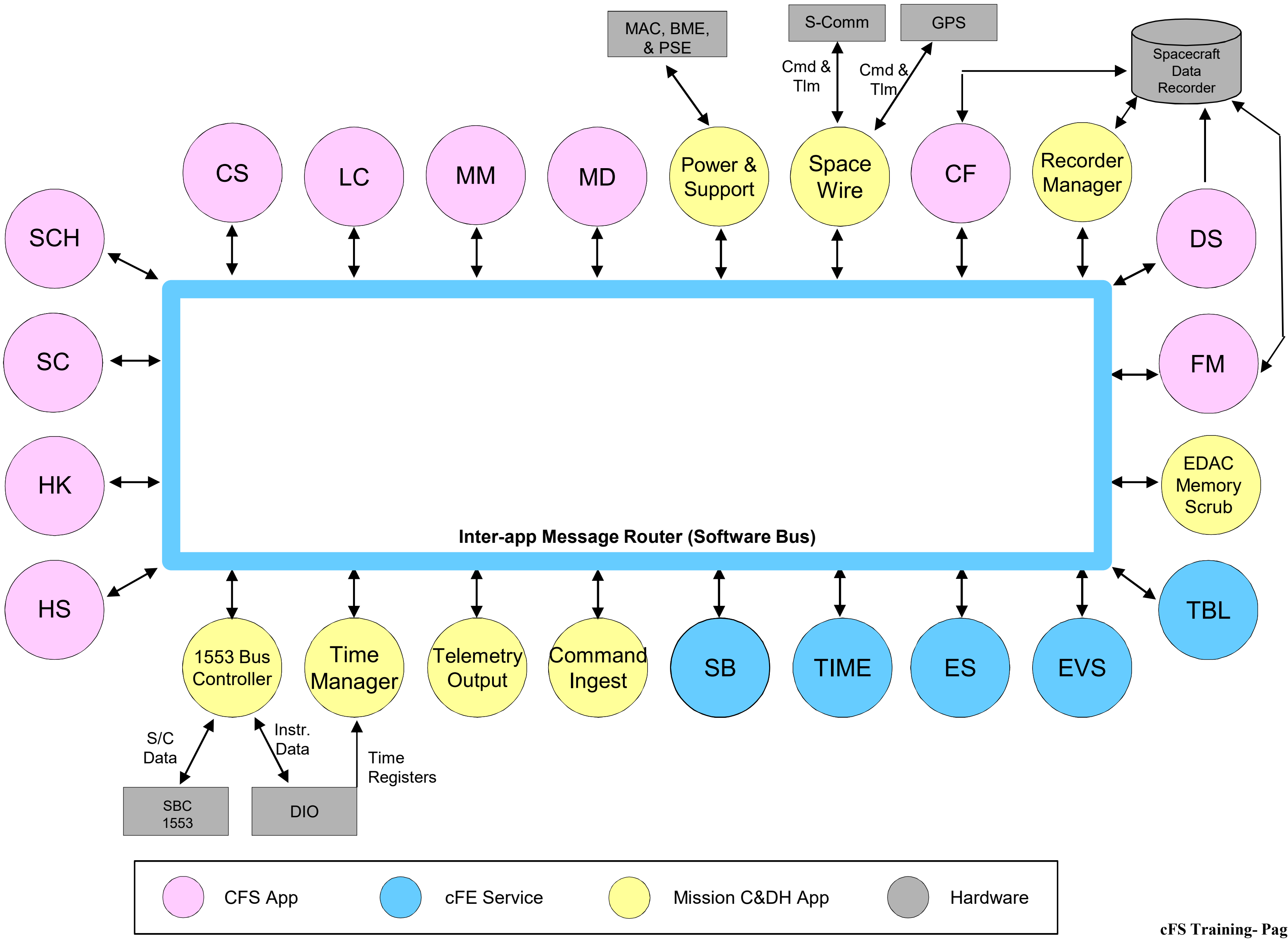
- **SCH, CI, and TO provide a runtime context that can be tailored for a particular environment**

- **Scheduler (SCH) App**
  - Sends software bus messages at pre-defined time intervals
  - Apps often use scheduled messages as wakeup signals

- **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

Mission Application Example



Existing Applications

GSFC Open Source Apps

Application	Function
CFDP	Transfers/receives file data to/from the ground
Checksum	Performs data integrity checking of memory, tables and files
Command Ingest Lab	Accepts CCSDS telecommand packets over a UDP/IP port
Data Storage	Records housekeeping, engineering and science data onboard for downlink
File Manager	Interfaces to the ground for managing files
Housekeeping	Collects and re-packages telemetry from other applications.
Health and Safety	Ensures critical tasks check-in, services watchdog, detects CPU hogging, calculates CPU utilization
Limit Checker	Provides the capability to monitor values and take action when exceed threshold
Memory Dwell	Allows ground to telemeter the contents of memory locations. Useful for debugging
Memory Manager	Provides the ability to load and dump memory
Software Bus Network	Passes Software Bus messages over various “plug-in” network protocols
Scheduler	Schedules onboard activities (e.g. HK requests)
Scheduler Lab	Simple activity scheduler with a one second resolution
Stored Command	Onboard Commands Sequencer (absolute and relative)
Stored Command Absolute	Allows concurrent processing of up to 5 (configurable) absolute time sequences
Telemetry Output Lab	Sends CCSDS telemetry packets over a UDP/IP port

Fault Detection and Correction Apps

- **Limit Checker (LC) – Monitors telemetry and responds to limit violations**

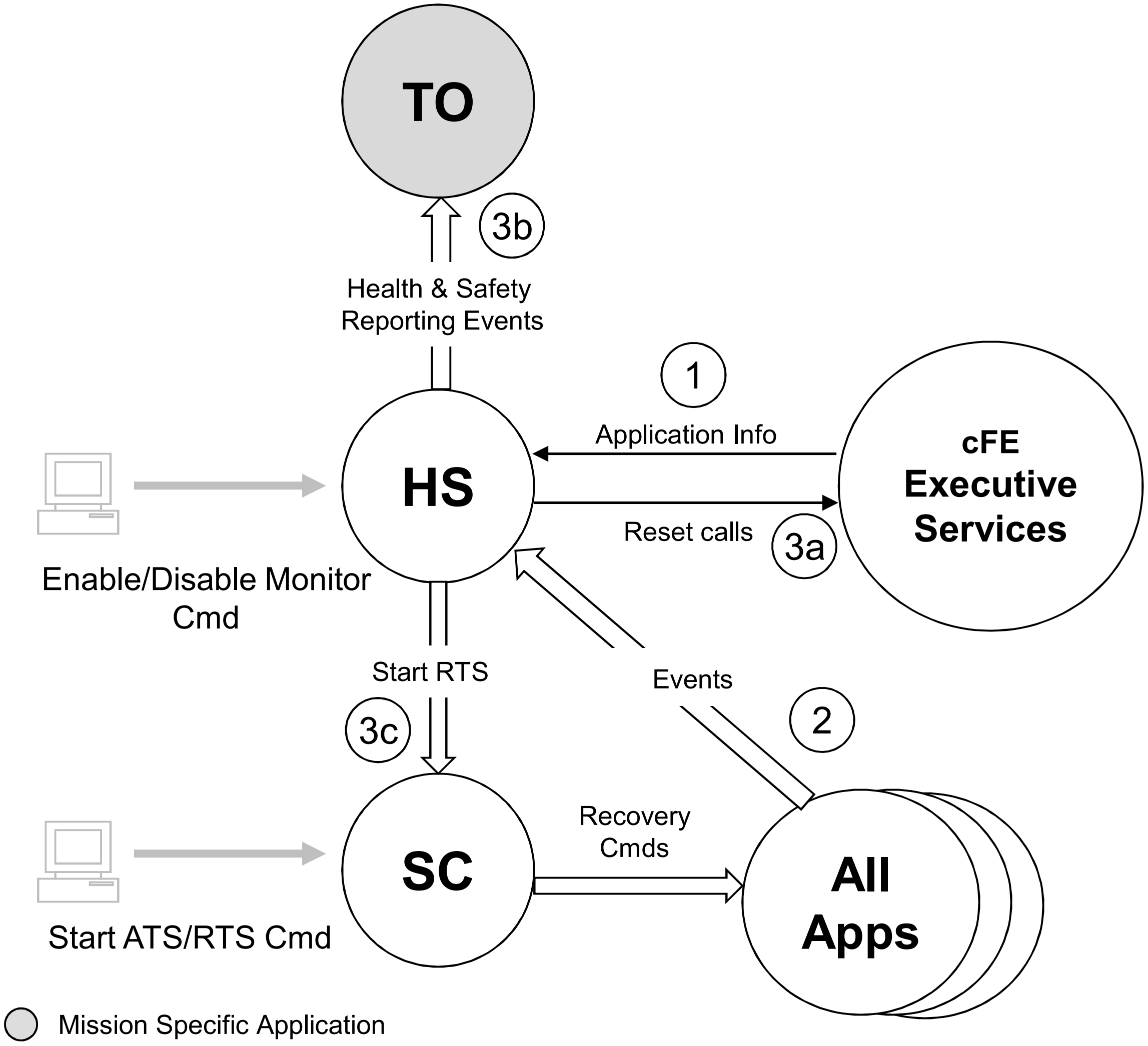
- **Health & Safety (HS) – Ensures critical tasks check-in, services watchdog, detects CPU hogging, calculates CPU utilization**

- **Checksum (CS) – Performs data integrity checking of memory, tables and files**

- **Stored Commands (SC) – Onboard commands sequencer (absolute and relative); used in combination with LC**

Operational Scenarios
Health & Safety

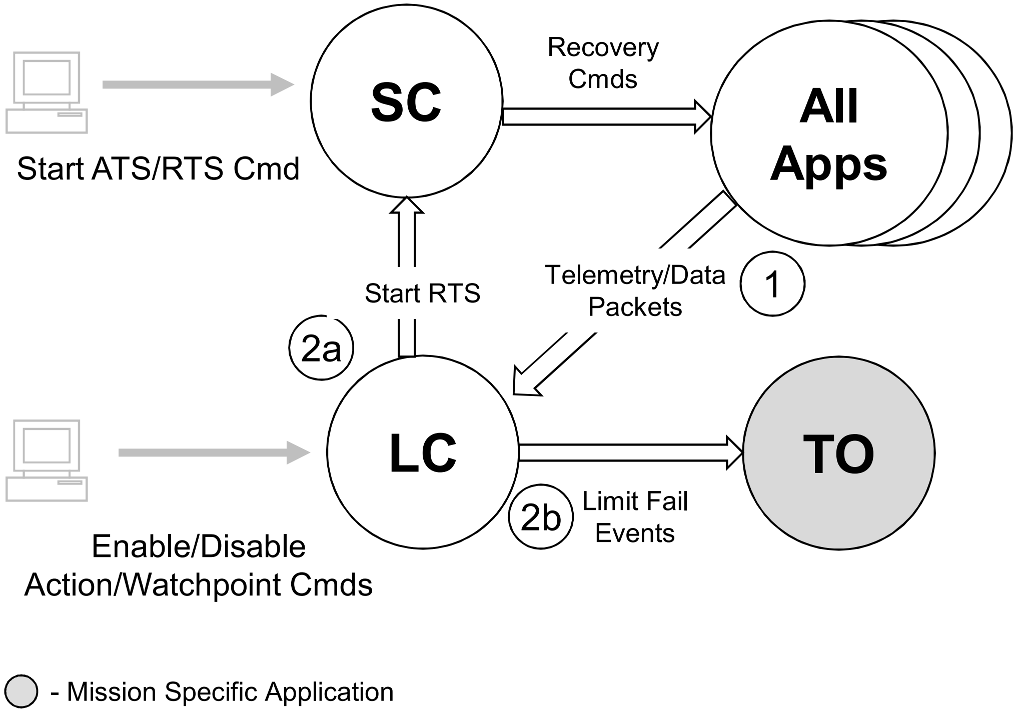
1) **HS monitors applications**
2) **HS monitors event messages**
3) **HS Table specified actions are taken in response to application and event monitoring:**
   a) **Reset applications or the processor**
   b) **Send Event message**
   c) **Initiate Stored Command (SC) recovery sequence**



Not pictured: HS manages watchdog, reports CPU utilization & detects hogging, and outputs aliveness heartbeat to UART.

Operational Scenarios
Fault Detection

1) LC monitors table specified telemetry and data (watchpoints)
2) LC evaluates actionpoints and takes action upon detected failure condition:
   a) Initiate Stored Command (SC) recovery sequence
   b) Send failure event messages

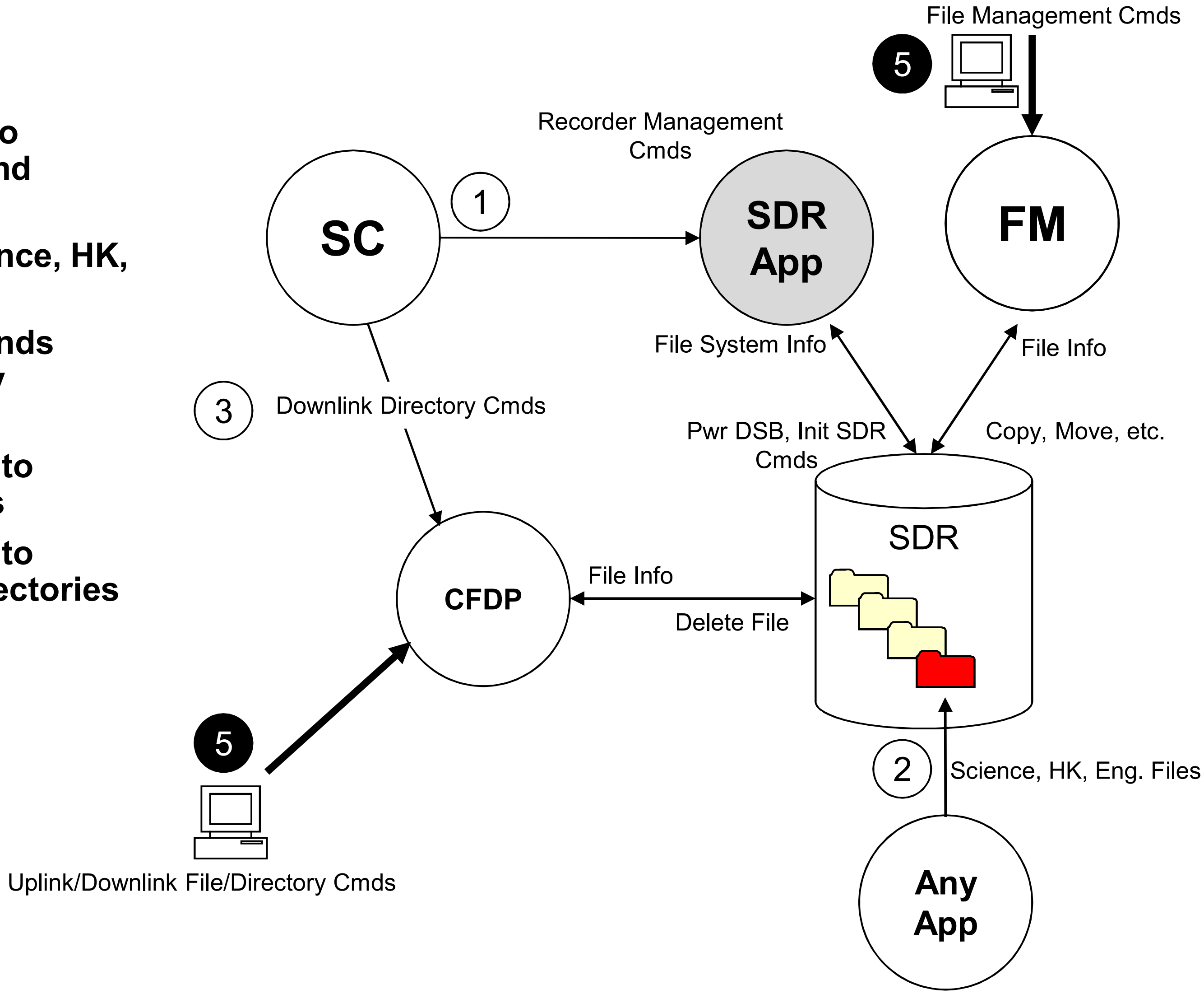


File & Data Management Apps

- **File Manager (FM) – Provides onboard file system operations**
- **Data Storage (DS) – Records housekeeping, engineering and science data onboard for downlink**
- **CFDP (CF) – Transfers/receives file data to/from the ground**
- **Housekeeping (HK) – Collects and re-packages telemetry from other applications**

Operational Scenarios
File Management

1) **Stored commands sent to initialize file system(s) and create partitions**
2) **Applications create Science, HK, and/or Engineering files**
3) **SC (typically via ATS) sends CFDP downlink directory commands**
4) **Ground commands sent to uplink and downlink files**
5) **Ground commands sent to manage the files and directories in the file system(s).**



🟥 - CFDP Hot Directory

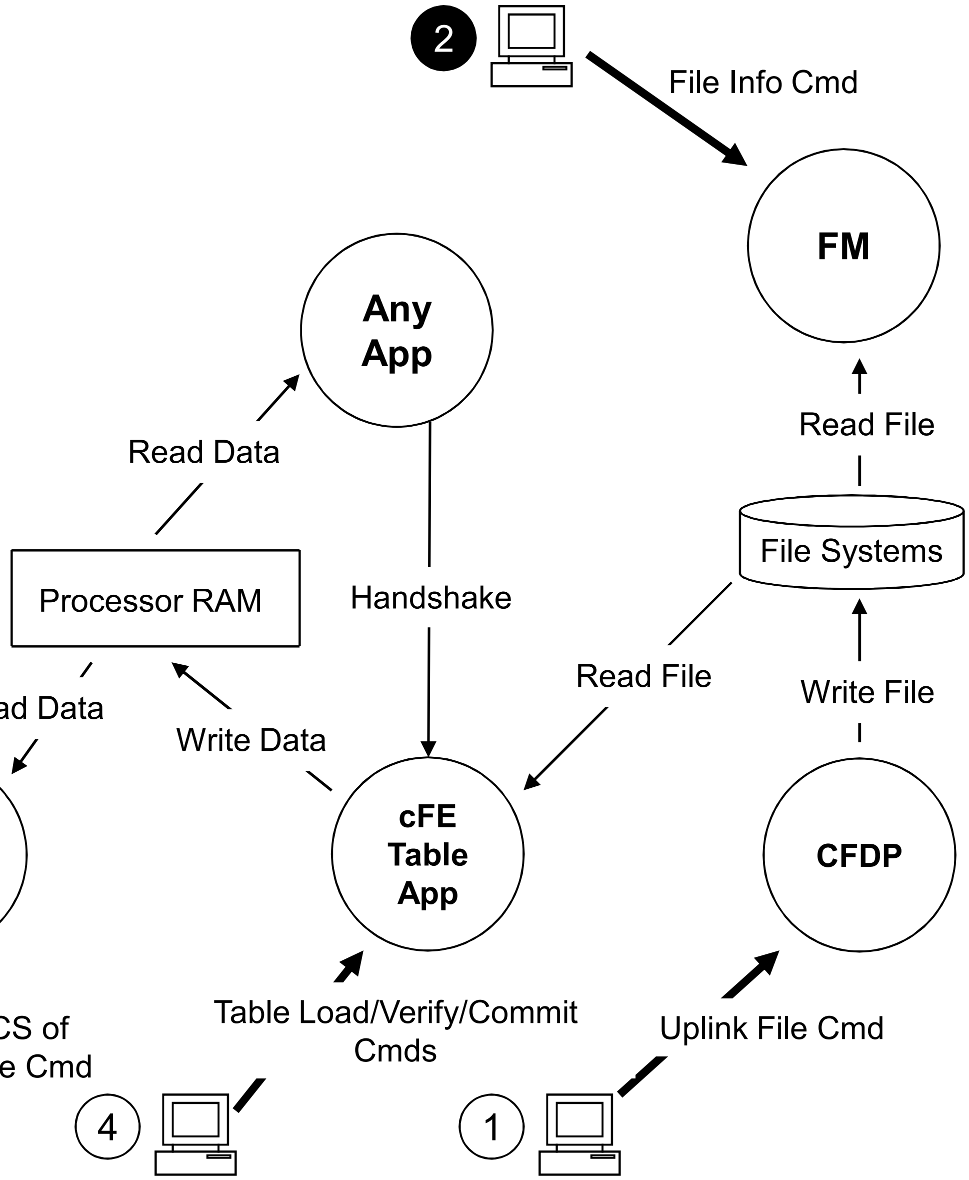
○ - Mission Specific Application

● - Optional Step

Operational Scenarios
Uplink System Tables

1) Uplink table – table is written to File System
2) Optionally CRC the table file (via FM file info command)
3) Disable background checksuming of the table
4) Send Table commands:
   - Load – reads table file and copies contents into active buffer
   - Validate – authenticates table data in the active buffer
   - Activate – writes/commits table data to RAM

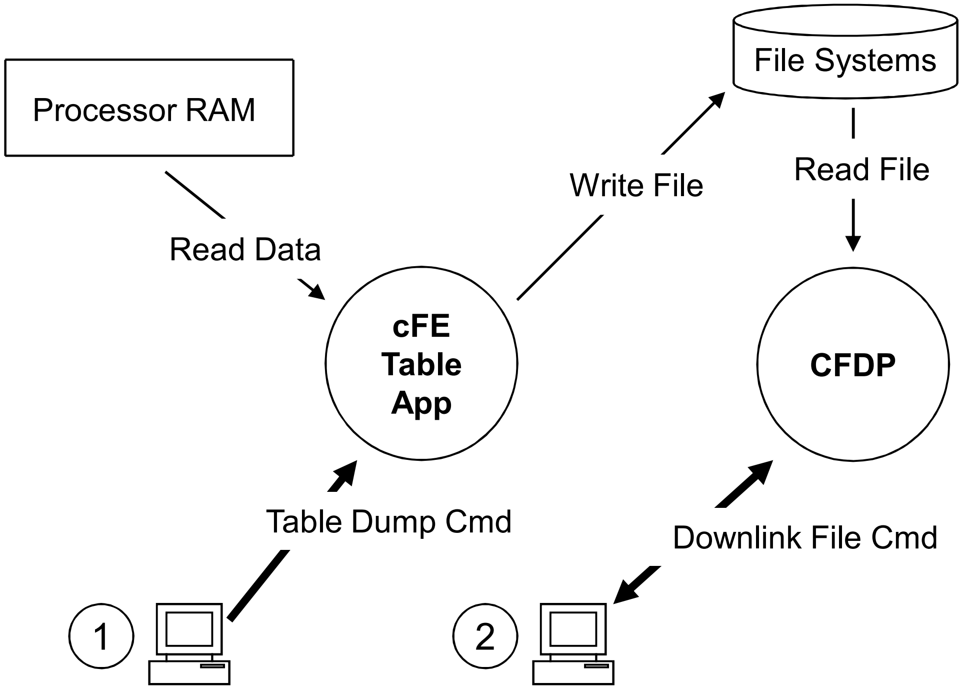
   Application handshakes with Table Services to read updated table data
5) Enable background checksumming of the table



● - Optional Step

Operational Scenarios
Dump System Tables

1) **Send Table dump command – table file is written to File System**
2) **Downlink file – table is written to ground File System.**



System Operations Applications

- **Scheduler (SCH) – Schedules onboard activities; many other applications depend on Scheduler**

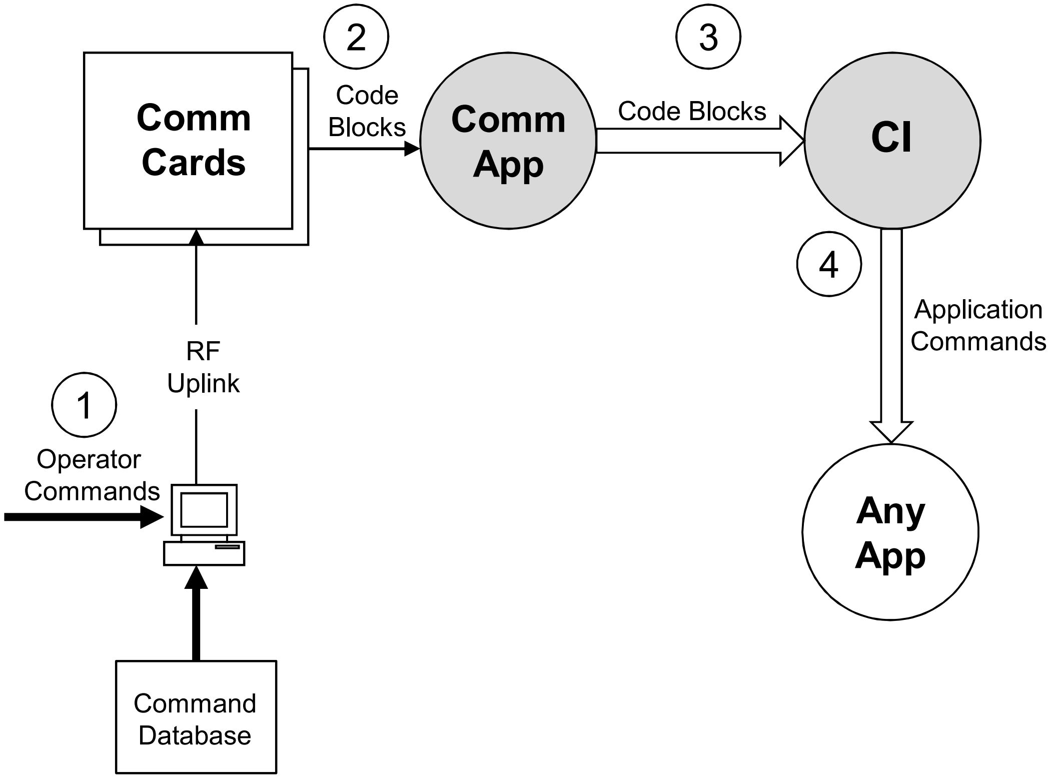
- **Command Ingest (CI) – Receives ground commands, validates them, and distributes them throughout the system; this app is often custom**

- **Telemetry Output (TO) – Downlinks telemetry; this app is often custom**

- **Stored Commands (SC) – Executes onboard command sequences (absolute and relative)**

Operational Scenarios
Uplink

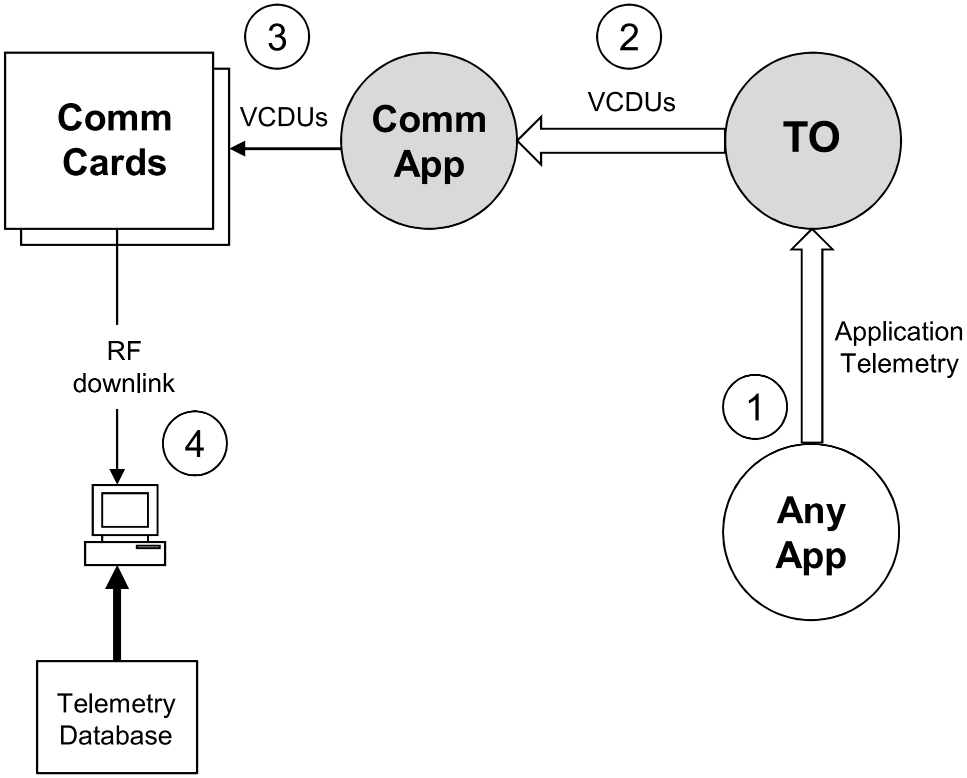
1) Commands sent from ground system are received by communication hardware
2) Communication hardware processes commands received and sends code blocks to receiving application.
3) Communication application strips off any hardware protocol wrappers, packages Code Blocks for transfer over software bus , and forwards Code Blocks to CI application
4) CI assembles command packets, performs command authentication, and sends commands to subscribed applications



● Mission Specific Application

Operational Scenarios
Telemetry Packet Downlink

1) Telemetry is collected from the various applications in the system and routed to TO application
2) TO collects, filters, and builds real-time VCDUs for downlink. The VCDU’s are packaged and routed over the software bus
3) Communication application strips off software bus headers, packages VCDUs in hardware protocol wrappers and outputs VCDUs across hardware link.
4) Telemetry is received by the ground system from communication hardware



● Mission Specific Application

Exercise 7 - Build applications

Part 1- Integrate the Scheduler application

1. Clone the Scheduler application

```
cd cFS/apps
git clone https://github.com/nasa/SCH.git sch
cd sch
git checkout rc-2.2.2
git pull
```

2. Replace "sch_lab" with "sch" in the sample_defs/targets.cmake file (line 88)

3. Update the cFE startup script (sample_defs/cpu1_cfe_es_startup.scr) by replacing sch_lab entry with:

```
CFE_APP, /cf/sch.so,         SCH_AppMain,     SCH,  80,   16384, 0x0, 0;
```

NOTE: Steps 2 and 3 (adding an app to the targets.cmake file and the startup script) can be repeated to add any app to the cFS build

NOTE: The sample_defs/cpu1_cfe_es_startup.scr file gets copied to the build directory and renamed to "cfe_es_startup.scr" during the "make install" part of the build process

Exercise 7 - Build applications

Part 1- Integrate the Scheduler application (Continued)

4. Update SCH table paths. In the apps/sch/fsw/platform_inc/sch_platform_cfg.h file, change the following #defines to the values shown below.

```
#define SCH_SCHEDULE_FILENAME  "/cf/sch_def_schtbl.tbl"
#define SCH_MESSAGE_FILENAME   "/cf/sch_def_msgtbl.tbl"
```

5. Build the cFS

```
make clean
make prep
make
make install
```

5. Run the cFE

```
cd build/exe/cpu1
./core-cpu1
```

Exercise 7 - Build applications

Part 1- Integrate the Scheduler application (Continued)

At this point you should see an error message that the SCH table could not be loaded.

```
      1980-012-14:03:20.25327 CFE_TBL:Load-App(8) Fail to load Tbl 'SCH.SCHED_DEF' from '/cf/sch_def_schtbl.tbl'
(Stat=0xFFFFFFFF)
      EVS Port1 42/1/CFE_TBL 93: SCH Failed to Load 'SCH.SCHED_DEF' from '/cf/sch_def_schtbl.tbl',
Status=0xFFFFFFFF
```

NOTE: The table name in the event message ("SCH.SCHED_DEF") includes the cFE name specified in the cfe_es_startup.scr file. The table name is specified in the table's source file. Mismatches between the table name in the source file and the app name in the startup script is a common source of errors.

Exercise 7 - Build applications

Part 1- Integrate the Scheduler application (Continued)

6. Fix the SCH CMakeLists.txt file by adding the following lines to the end of the file apps/sch/CMakeLists.txt

```
include_directories(fsw/src)
aux_source_directory(fsw/tables APP_TABLE_FILES)
add_cfe_tables(sch ${APP_TABLE_FILES})
```

NOTE: The "add_cfe_tables" call must always come after the "add_cfe_app" call in the CMakeLists.txt file

7. Build the cFS

```
make clean
make prep
make
make install
```

8. Run the cFE

```
cd build/exe/cpu1
./core-cpu1
```

Exercise 7 - Build applications

Part 2- Configure SCH to command the sample_app

1. Navigate to the apps/sch/fsw/tables directory
2. Open sch_def_msgtbl.c
3. Add an include statement for sample_app_msgids.h

```
    #include sample_app_msgids.h
```

4. Replace the line for Command Id #6 with the following

```
        { { CFE_MAKE_BIG16(SAMPLE_APP_CMD_MID), CFE_MAKE_BIG16(0xC000),
CFE_MAKE_BIG16(0x0001), 0x0000 } },
```

The above line describes a no-operation command to sample_app. The first 3 fields are the CCSDS header. The fourth field is the command code (0 is the standard command code for a no-op command).

5. Save and close sch_def_msgtbl.c
6. Open sch_def_schtbl.c
7. Replace the first entry under Slot #1 with the following

```
        {  SCH_ENABLED,   SCH_ACTIVITY_SEND_MSG,      3,  0, 6,  SCH_GROUP_NONE},
```

The above line indicates that Command Id #6 (defined in step 4) should be sent every 3 seconds.

Exercise 7 - Build applications

Part 2- Configure SCH to command the sample_app (continued)

8. Add the following line to the scheduler CMakeLists.txt file before the "add_cfe_app" function call.

```
        include_directories(${sample_app_MISSION_DIR}/fsw/platform_inc)
```

The above line will allow the sch app to successfully find the sample_app_msgids.h file added in Step 3.

9. Rebuild the cFS.

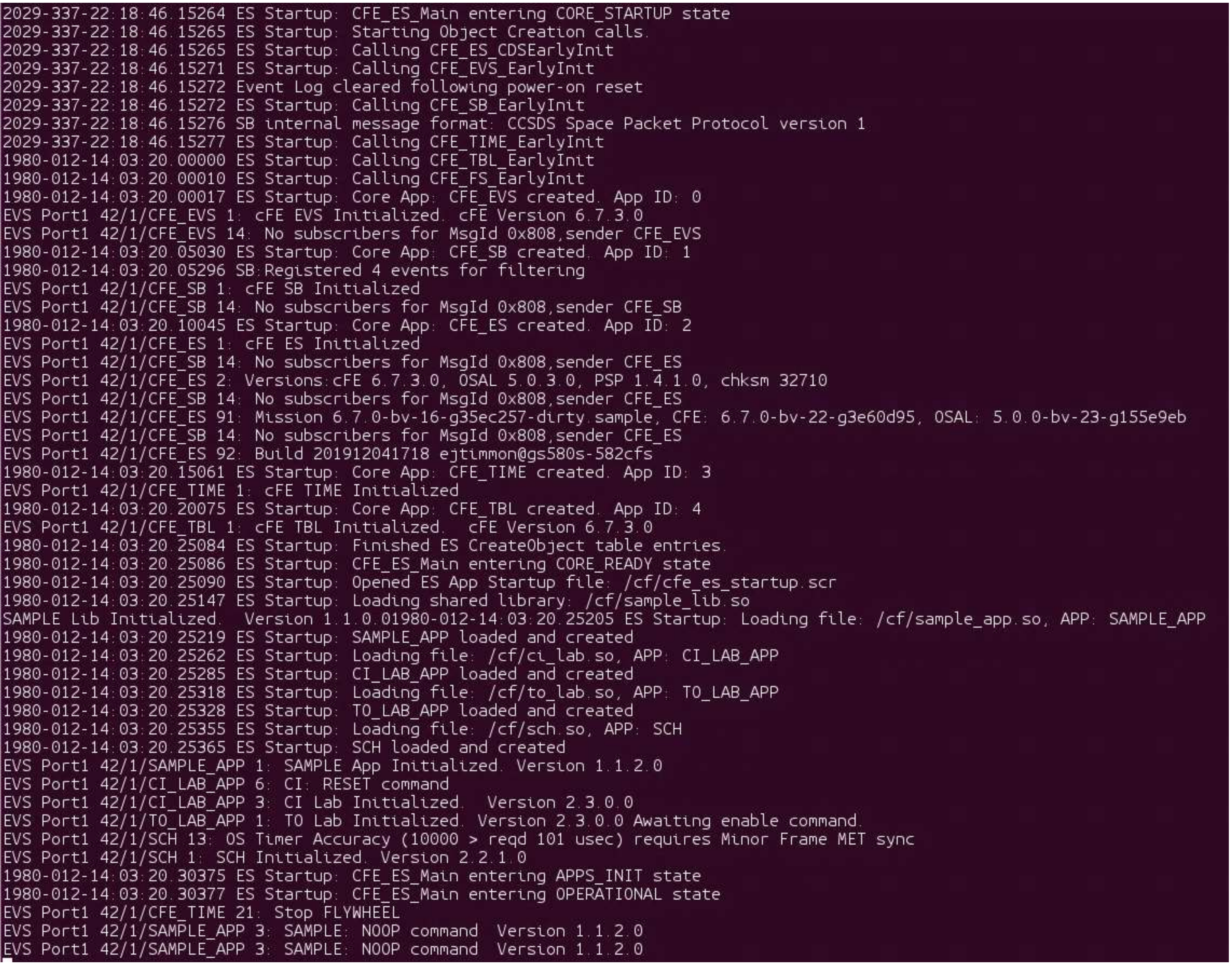
```
        make clean
        make prep
        make
        make install
```

10. Run the cFE

```
        cd build/exe/cpu1
        ./core-cpu1
```

NOTE: The process just completed is the same process that can be used to add housekeeping requests and wakeup messages to the scheduler application

Exercise 7 Recap



```
2029-337-22:18:46.15264 ES Startup: CFE_ES_Main entering CORE_STARTUP state
2029-337-22:18:46.15265 ES Startup: Starting Object Creation calls.
2029-337-22:18:46.15265 ES Startup: Calling CFE_ES_CDSEarlyInit
2029-337-22:18:46.15271 ES Startup: Calling CFE_EVS_EarlyInit
2029-337-22:18:46.15272 Event Log cleared following power-on reset
2029-337-22:18:46.15272 ES Startup: Calling CFE_SB_EarlyInit
2029-337-22:18:46.15276 SB internal message format: CCSDS Space Packet Protocol version 1
2029-337-22:18:46.15277 ES Startup: Calling CFE_TIME_EarlyInit
1980-012-14:03:20.00000 ES Startup: Calling CFE_TBL_EarlyInit
1980-012-14:03:20.00010 ES Startup: Calling CFE_FS_EarlyInit
1980-012-14:03:20.00017 ES Startup: Core App: CFE_EVS created. App ID: 0
EVS Port1 42/1/CFE_EVS 1: cFE EVS Initialized. cFE Version 6.7.3.0
EVS Port1 42/1/CFE_EVS 14: No subscribers for MsgId 0x808,sender CFE_EVS
1980-012-14:03:20.05030 ES Startup: Core App: CFE_SB created. App ID: 1
1980-012-14:03:20.05296 SB:Registered 4 events for filtering
EVS Port1 42/1/CFE_SB 1: cFE SB Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_SB
1980-012-14:03:20.10045 ES Startup: Core App: CFE_ES created. App ID: 2
EVS Port1 42/1/CFE_ES 1: cFE ES Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 2: Versions:cFE 6.7.3.0, OSAL 5.0.3.0, PSP 1.4.1.0, chksm 32710
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 91: Mission 6.7.0-bv-16-g35ec257-dirty.sample, CFE: 6.7.0-bv-22-g3e60d95, OSAL: 5.0.0-bv-23-g155e9eb
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 92: Build 201912041718 ejtimmon@gs580s-582cfs
1980-012-14:03:20.15061 ES Startup: Core App: CFE_TIME created. App ID: 3
EVS Port1 42/1/CFE_TIME 1: cFE TIME Initialized
1980-012-14:03:20.20075 ES Startup: Core App: CFE_TBL created. App ID: 4
EVS Port1 42/1/CFE_TBL 1: cFE TBL Initialized.  cFE Version 6.7.3.0
1980-012-14:03:20.25084 ES Startup: Finished ES CreateObject table entries.
1980-012-14:03:20.25086 ES Startup: CFE_ES_Main entering CORE_READY state
1980-012-14:03:20.25090 ES Startup: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.25147 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized.  Version 1.1.0.01980-012-14:03:20.25205 ES Startup: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25219 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25262 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25285 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25318 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25328 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25355 ES Startup: Loading file: /cf/sch.so, APP: SCH
1980-012-14:03:20.25365 ES Startup: SCH loaded and created
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable command.
EVS Port1 42/1/SCH 13: OS Timer Accuracy (10000 > reqd 101 usec) requires Minor Frame MET sync
EVS Port1 42/1/SCH 1: SCH Initialized. Version 2.2.1.0
1980-012-14:03:20.30375 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30377 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/SAMPLE_APP 3: SAMPLE: NOOP command  Version 1.1.2.0
EVS Port1 42/1/SAMPLE_APP 3: SAMPLE: NOOP command  Version 1.1.2.0
```

SCH instead of SCH_lab

No-op messages

Application Design

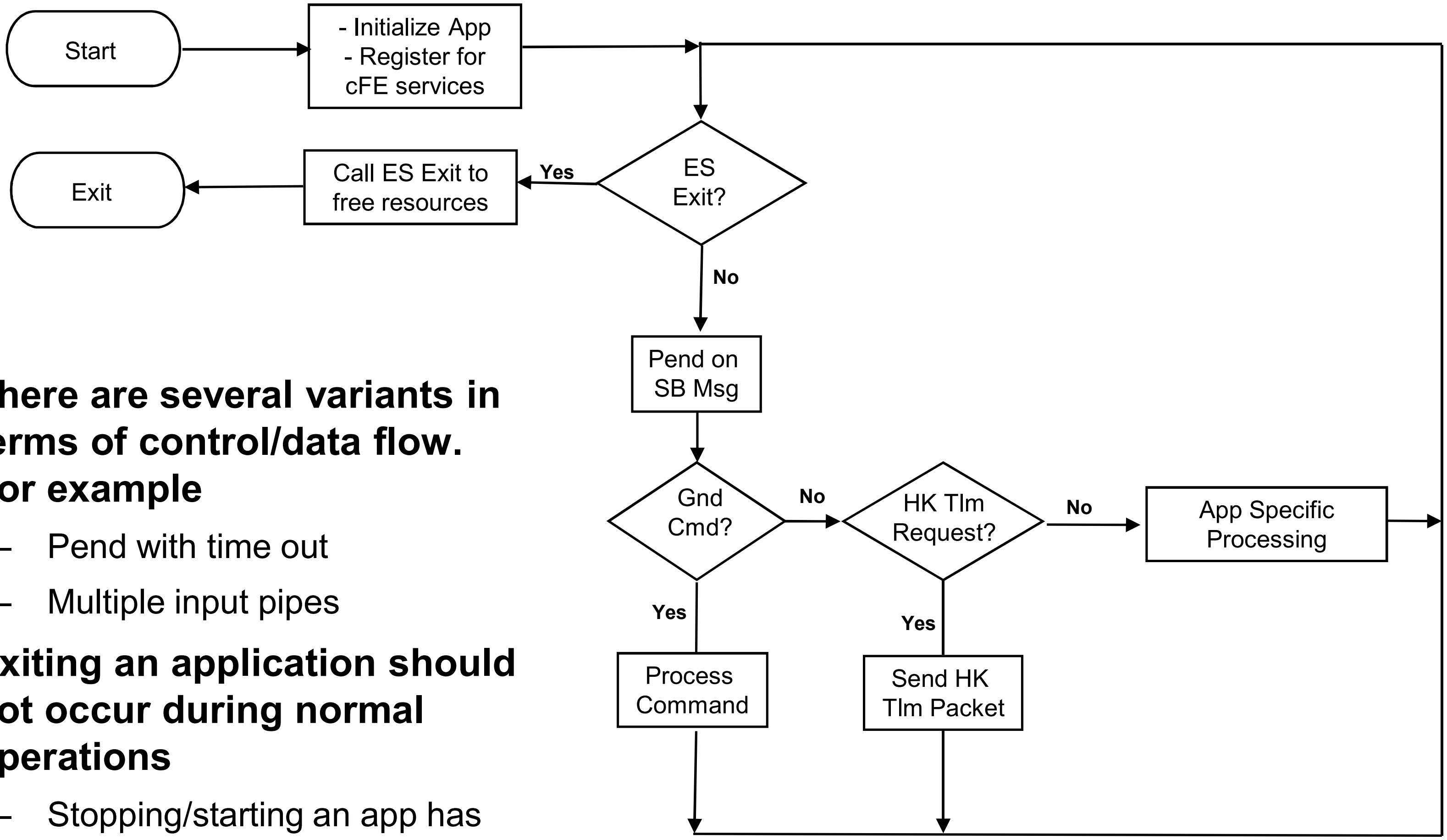
Application Design Resources

- **cFE/docs/cFE Application Developers Guide.doc**
  - Provides a good description of how to use cFE services/features
  - Provides one example of an application template
- **sample_app**
  - Provides an operational example of a basic application
  - https://github.com/nasa/sample_app/
- **Application frameworks**
  - Organizations have created frameworks in C and C++ but they are not publically available
- **“Hello World” app generation tools**
  - Multiple tools exist, but none have been sanctioned as demonstrating best practices
- **Application design patterns**
  - There are patterns but they have not been formally captured
  - When creating a new app look for an existing app that has similar operational context

Application Design Practices

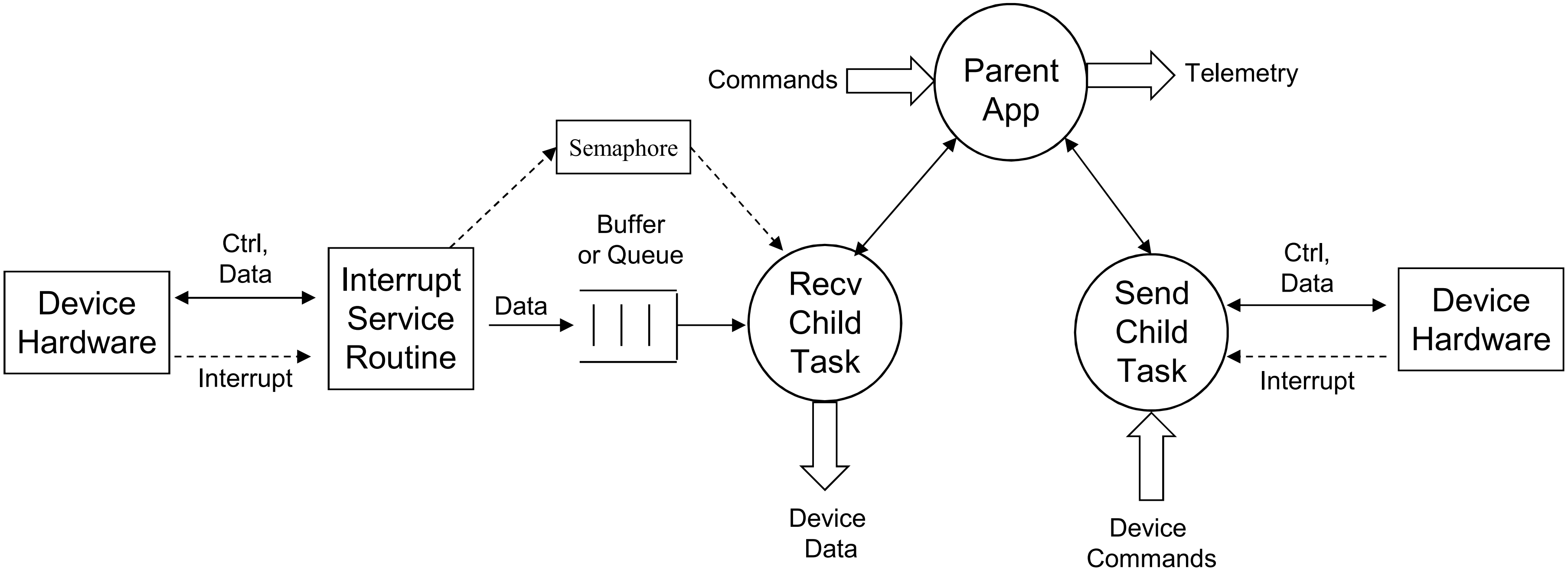
- **Allocate resources during initialization to help keep run loop deterministic**
- **Use a lower priority child task for long operations like a memory dump**
  - Create child tasks during initialization
- **Register with EVS immediately after registering app so local event log can be used instead of system log**
- **NOOP command sends an informational event message with app's version number**
- **Use SCH app to periodically send a "send housekeeping" message**
  - Housekeeping data includes command counters and general app status
  - 3 to 5 seconds is a common interval
  - Attitude Determination and Control apps don't typically use this pattern

Generic App Design



- **There are several variants in terms of control/data flow. For example**
  - Pend with time out
  - Multiple input pipes
- **Exiting an application should not occur during normal operations**
  - Stopping/starting an app has been used for in-orbit maintenance

I/O Application Design Pattern



- **General control/data conceptual flow**
  - Each communication bus has a specific protocol
- **Architectural role**
  - Read device data and publish on software bus
  - Receive software bus messages and send to the device

Exercise 8 - Add a command to sample_app

Part 1 – Add new command code event message

1. Navigate the the sample_app source directory

```
        cd apps/sample_app/fsw/src
```

2. Open the sample_app_msg.h file and add a new command code

```
        #define SAMPLE_APP_HELLO_WORLD_CC          3
```

3. Open the sample_app_events.h file and add a new event message and update the number of events.

```
        #define SAMPLE_HELLO_WORLD_INF_EID            8
        #define SAMPLE_EVENT_COUNTS                   8
```

4. Open the sample_app.c file and add the new event message to the event filter set up in SAMPLE_AppInit

```
        Sample_AppData.SAMPLE_EventFilters[7].EventID = SAMPLE_HELLO_WORLD_INF_EID;
        Sample_AppData.SAMPLE_EventFilters[7].Mask    = 0x0000;
```

Exercise 8 - Add a command to sample_app

Part 2 – Add code to handle new command

5. Add a case for the new command code in SAMPLE_ProcessGroundCommand

```
case SAMPLE_APP_HELLO_WORLD_CC:
    if (SAMPLE_VerifyCmdLength(Msg, sizeof(SAMPLE_Noop_t))) {
        SAMPLE_HelloCmd((SAMPLE_Noop_t * )Msg);
    }
    break;
```

6. Add a new function called SAMPLE_HelloCmd

```
void SAMPLE_HelloCmd( const SAMPLE_Noop_t * Msg )  {
    Sample_AppData.CmdCounter++;

    CFE_EVS_SendEvent(SAMPLE_HELLO_WORLD_INF_EID, CFE_EVS_INFORMATION,
                      "Hello, World.  This is sample_app!");
    return;
}
```

7. Add a function prototype for the new function in sample_app.h

```
void  SAMPLE_HelloCmd(const SAMPLE_Noop_t * Msg);
```

Exercise 8 - Add a command to sample_app

Part 3 – Add new command to scheduler

8. Edit the SCH configuration to send the Hello command instead of a No-Op. Open apps/sch/fsw/tables/sch_def_msgtbl.c and modify Command Id #6 to the following line

```
        { { CFE_MAKE_BIG16(SAMPLE_APP_CMD_MID), CFE_MAKE_BIG16(0xC000),
CFE_MAKE_BIG16(0x0001), CFE_MAKE_BIG16(0x0003)} },
```

In the above line, the command code is changed to 3 to match the command code defiend in Step 3

9. Rebuild the cFS.

```
        make clean
        make prep
        make
        make install
```

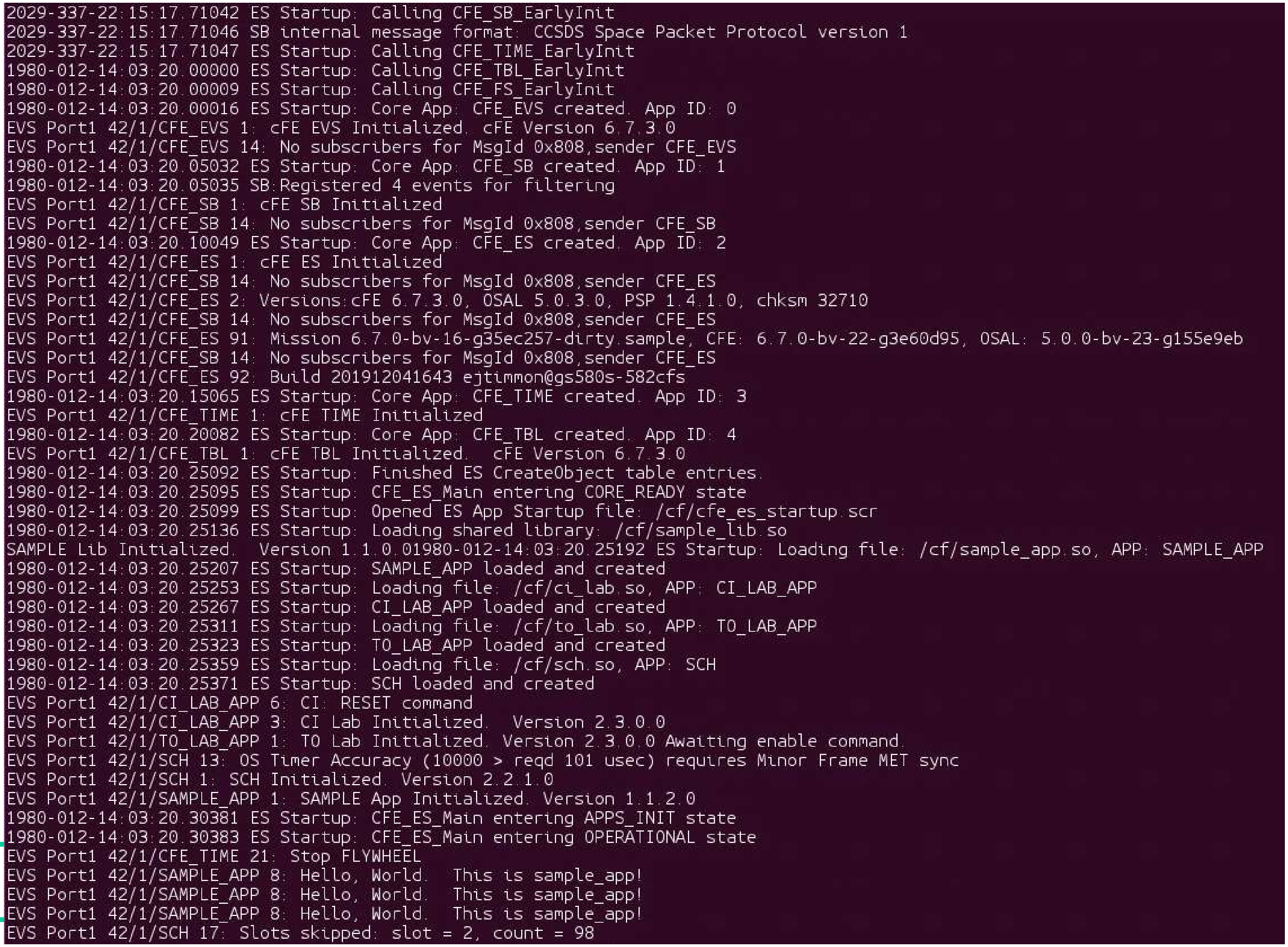
10. Run the cFE

```
        cd build/exe/cpu1
        ./core-cpu1
```

"Hello World" messages should now be appearing regularly

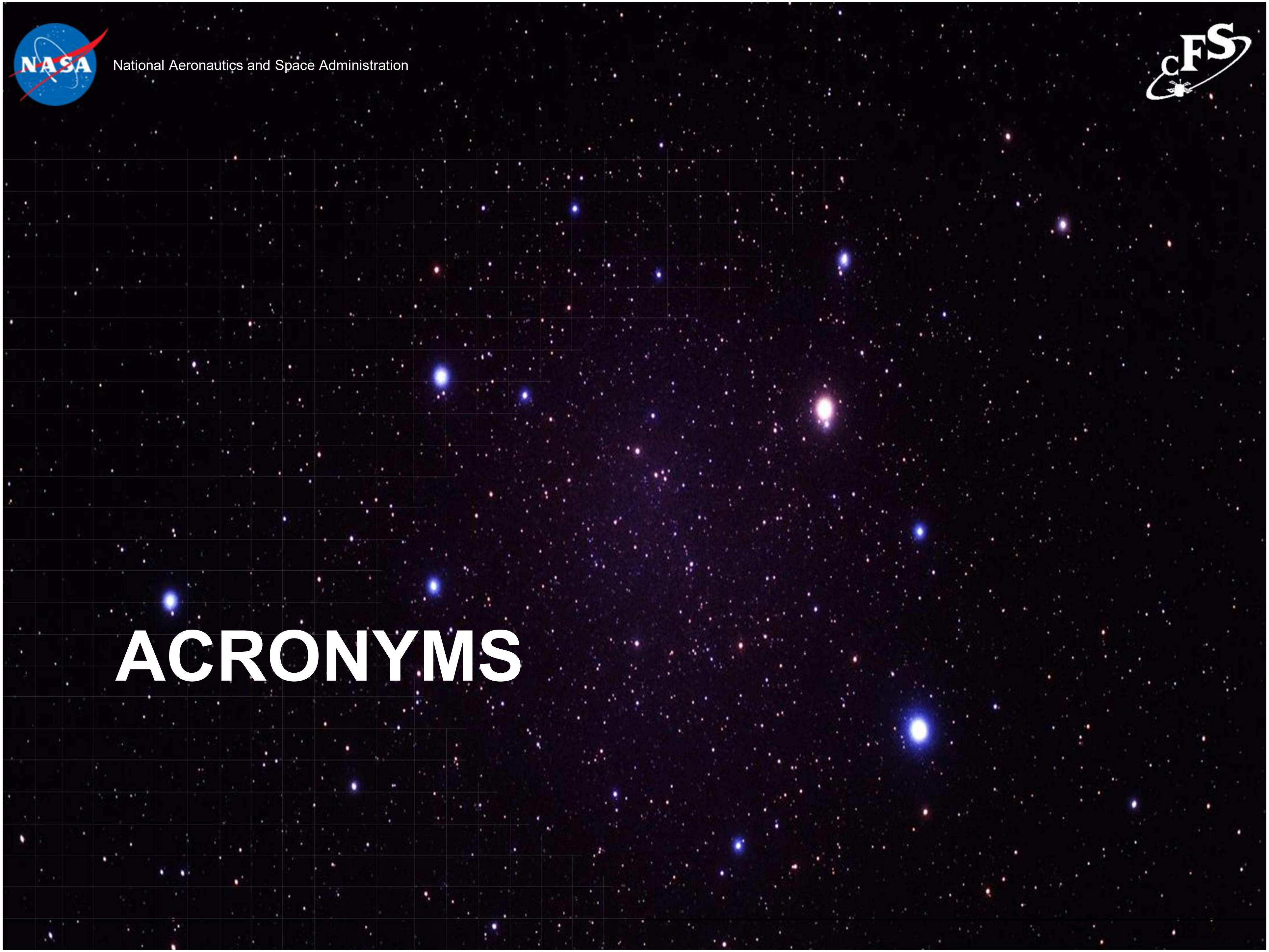
NOTE: In the above process, Steps 1-7 describe the general process for adding any command to an application.

Exercise 8 Recap



```
2029-337-22:15:17.71042 ES Startup: Calling CFE_SB_EarlyInit
2029-337-22:15:17.71046 SB internal message format: CCSDS Space Packet Protocol version 1
2029-337-22:15:17.71047 ES Startup: Calling CFE_TIME_EarlyInit
1980-012-14:03:20.00000 ES Startup: Calling CFE_TBL_EarlyInit
1980-012-14:03:20.00009 ES Startup: Calling CFE_FS_EarlyInit
1980-012-14:03:20.00016 ES Startup: Core App: CFE_EVS created. App ID: 0
EVS Port1 42/1/CFE_EVS 1: cFE EVS Initialized. cFE Version 6.7.3.0
EVS Port1 42/1/CFE_EVS 14: No subscribers for MsgId 0x808,sender CFE_EVS
1980-012-14:03:20.05032 ES Startup: Core App: CFE_SB created. App ID: 1
1980-012-14:03:20.05035 SB:Registered 4 events for filtering
EVS Port1 42/1/CFE_SB 1: cFE SB Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_SB
1980-012-14:03:20.10049 ES Startup: Core App: CFE_ES created. App ID: 2
EVS Port1 42/1/CFE_ES 1: cFE ES Initialized
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 2: Versions:cFE 6.7.3.0, OSAL 5.0.3.0, PSP 1.4.1.0, chksm 32710
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 91: Mission 6.7.0-bv-16-g35ec257-dirty.sample, CFE: 6.7.0-bv-22-g3e60d95, OSAL: 5.0.0-bv-23-g155e9eb
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 92: Build 201912041643 ejtimmon@gs580s-582cfs
1980-012-14:03:20.15065 ES Startup: Core App: CFE_TIME created. App ID: 3
EVS Port1 42/1/CFE_TIME 1: cFE TIME Initialized
1980-012-14:03:20.20082 ES Startup: Core App: CFE_TBL created. App ID: 4
EVS Port1 42/1/CFE_TBL 1: cFE TBL Initialized.  cFE Version 6.7.3.0
1980-012-14:03:20.25092 ES Startup: Finished ES CreateObject table entries.
1980-012-14:03:20.25095 ES Startup: CFE_ES_Main entering CORE_READY state
1980-012-14:03:20.25099 ES Startup: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.25136 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized.  Version 1.1.0.01980-012-14:03:20.25192 ES Startup: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25207 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25253 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.25267 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25311 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25323 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.25359 ES Startup: Loading file: /cf/sch.so, APP: SCH
1980-012-14:03:20.25371 ES Startup: SCH loaded and created
EVS Port1 42/1/CI_LAB_APP 6: CI: RESET command
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized.  Version 2.3.0.0
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. Version 2.3.0.0 Awaiting enable command.
EVS Port1 42/1/SCH 13: OS Timer Accuracy (10000 > reqd 101 usec) requires Minor Frame MET sync
EVS Port1 42/1/SCH 1: SCH Initialized. Version 2.2.1.0
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Version 1.1.2.0
1980-012-14:03:20.30381 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.30383 ES Startup: CFE_ES_Main entering OPERATIONAL state
EVS Port1 42/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 42/1/SAMPLE_APP 8: Hello, World.  This is sample_app!
EVS Port1 42/1/SAMPLE_APP 8: Hello, World.  This is sample_app!
EVS Port1 42/1/SAMPLE_APP 8: Hello, World.  This is sample_app!
EVS Port1 42/1/SCH 17: Slots skipped: slot = 2, count = 98
```

New Hello
World
messages



ACRONYMS

Acronyms

Acronym	Definition	Acronym	Definition
API	Application Programmer Interface	CM	Configuration Management
APID	Application Process ID	CMD	Command
ATS	Absolute Time Sequence	COTS	Commercial Off The Shelf
BC	Bus Controller	CRC	Cyclic Redundancy Check
BSP	Board Support Package	CS	Checksum
C&DH	Command and Data Handling	DS	Data Storage
CCSDS	Consultative Committee for Space Data Systems	EEPROM	Electrically Erasable Programmable Read-Only Memory
CDS	Critical Data Store	ES	Executive Services
CESE	Center for Experimental Software Engineering	EVS	Event Services
CFDP	CCSDS File Delivery Protocol	FDC	Failure Detection and Correction
cFE	Core Flight Executive	FDIR	Failure Detection, Isolation, and Recovery
cFS	Core Flight Software System	FM	File Management, Fault Management

Acronyms

Acronym	Definition	Acronym	Definition
FSW	Flight Software	ITC	Independent Test Capability
GNC	Guidance Navigation and Control	ITOS	Integration Test and Operations System
GSFC	Goddard Space Flight Center	IV&V	Independent Verification and Validation
GOTS	Government Off The Shelf	LC	Limit Checker
GPM	Global Precipitation Measurement	Mbps	Megabits-per seconds
GPS	Global Positioning System	MD	Memory Dwell
Hi-Fi	High-Fidelity Simulation	MET	Mission Elapsed Timer
HK	Housekeeping	MM	Memory Manager
HS	Health & Safety	MS	Memory Scrub
HW	Hardware	NACK	Negative-acknowledgement
Hz	Hertz	NASA	National Aeronautics Space Agency
ITAR	International Traffic in Arms Regulations	NOOP	No Operation
ISR	Interrupt Service Routine	OS	Operating System

Acronyms

Acronym	Definition	Acronym	Definition
OSAL	Operating System Abstraction Layer	SC	Stored Command
PSP	Platform Support Package	SCH	Scheduler
PROM	Programmable Read-Only Memory	S-COMM	S-Band Communication Card
RAM	Random-Access Memory	SDR	Spacecraft Data Recorder
RT	Remote Terminal	SpW	Spacewire
R/T	Real-time	STCF	Spacecraft Time Correlation Factor
RTEMS	Real-Time Executive for Multiprocessor Systems (an RTOS)	SW	Software, Spacewire
RTOS	Real-Time Operating System	TAI	International Atomic Time
RTS	Relative Time Sequence	TBD	To be determined
SARB	Software Architecture Review Board	TBL	Table Services
S/C	Spacecraft	TLM	Telemetry
SB	Software Bus	TO	Telemetry Output
SBC	Single-Board Computer	UART	Universal Asynchronous Receiver/Transmitter

Acronyms

Acronym	Definition	Acronym	Definition
UDP	User Datagram Protocol	UTC	Coordinated Universal Time
UT	Unit Test	VCDU	Virtual Channel Data Unit