NASA/TM–20205000691/REV 2

Core Flight System (cFS)Training

cFS Caelum

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

October 2021

NASA/TM–20205000691/REV 2

Core Flight System (cFS)Training

cFS Caelum

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

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

October 2021

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

Core Flight System (cFS) Training

cFS Caelum

Module 1: Introduction

Course Agenda

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

Course Audience & Prerequisites

- **Audience: 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 3.8, PyQt5, PyZMQ

Course Learning Objectives

- Understand the architecture of the cFS
- Build and execute the cFS
- Interact with the cFS through a ground system
- Modify a cFS application

Introduction Agenda

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

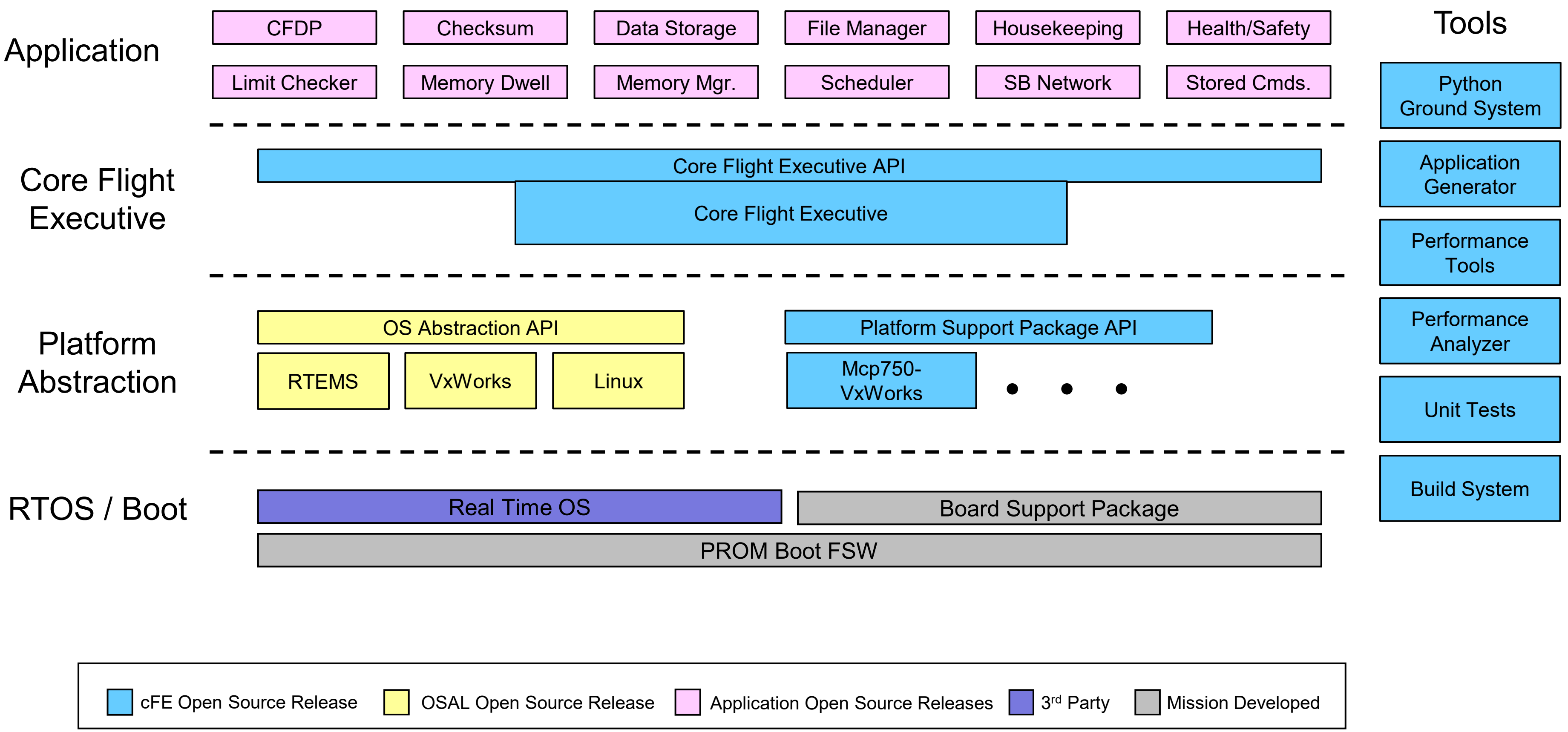
What is cFS?

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, Checksum, Data Storage, File Manager, Housekeeping, Health/Safety

Limit Checker, Memory Dwell, Memory Mgr., Scheduler, SB Network, Stored Cmds.

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

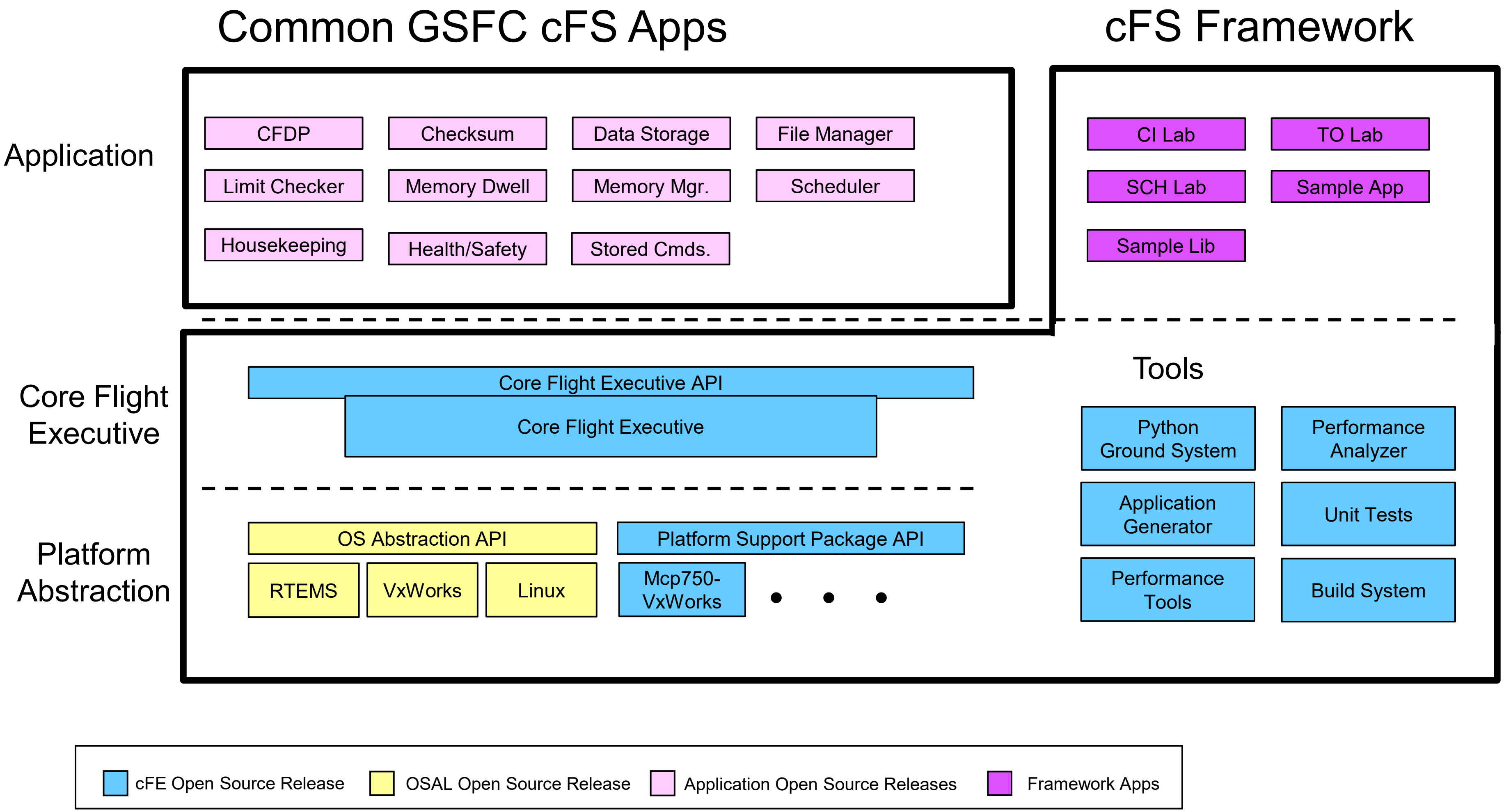
Performance Analyzer

Unit Tests

Build System

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

cFS Organization



Common GSFC cFS Apps

cFS Framework

Application

- CFDP
- Checksum
- Data Storage
- File Manager
- Limit Checker
- Memory Dwell
- Memory Mgr.
- Scheduler
- Housekeeping
- Health/Safety
- Stored Cmds.

- CI Lab
- TO Lab
- SCH Lab
- Sample App
- Sample Lib

Core Flight Executive

- Core Flight Executive API
- Core Flight Executive

Platform Abstraction

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

Tools

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

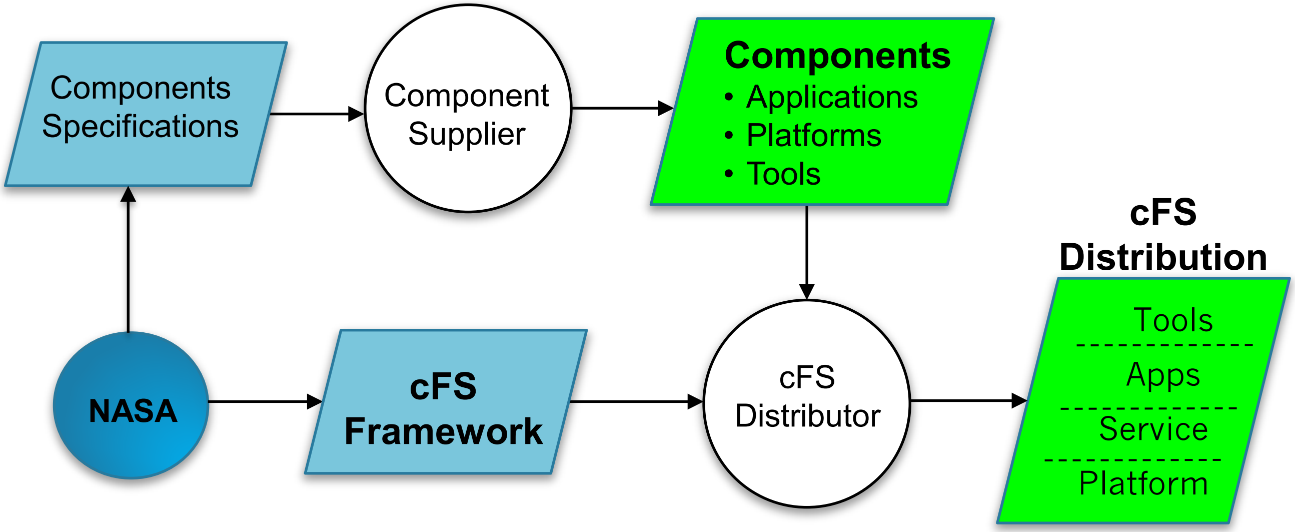
🟦 cFE Open Source Release 🟨 OSAL Open Source Release ◻ Application Open Source Releases 🟪 Framework Apps

Key Definitions

- **Framework – The set of individual services, applications, tools, and infrastructure supported by the open source community Configuration Control Board (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

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"

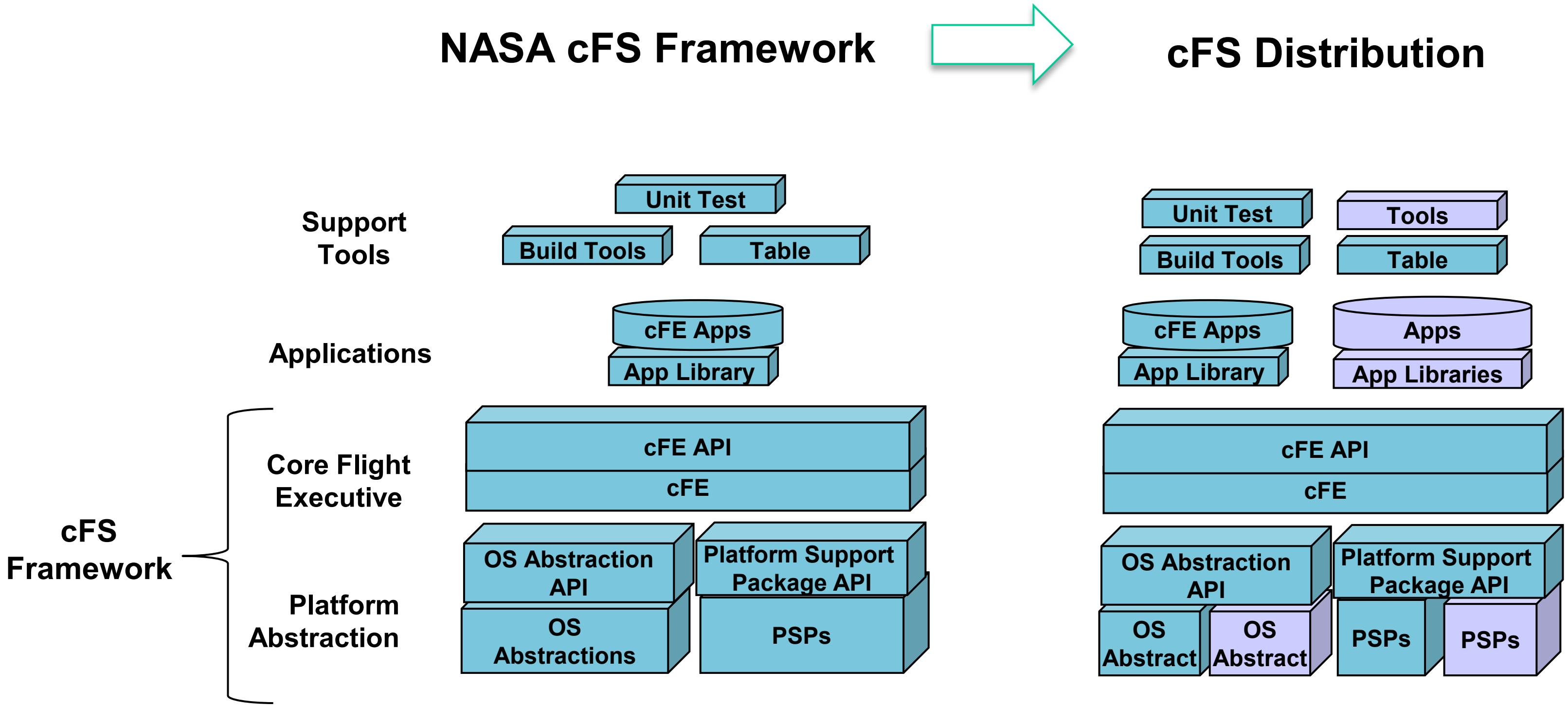
- **cFS Bundle**
  - Contains the cFS Framework packaged with additional components to create a system that can easily be built, executed, and unit tested on a Linux platform
  - http://github.com/nasa/cFS

- **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)
  - 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**
  - Main Branch – always has the latest code
  - Integration Candidates – updated after the weekly CCB meeting
  - Release Candidates – periodically tagged from master

- **User Contributions**
  - A Contributor License Agreement (CLA) is required for each contributor to the open source

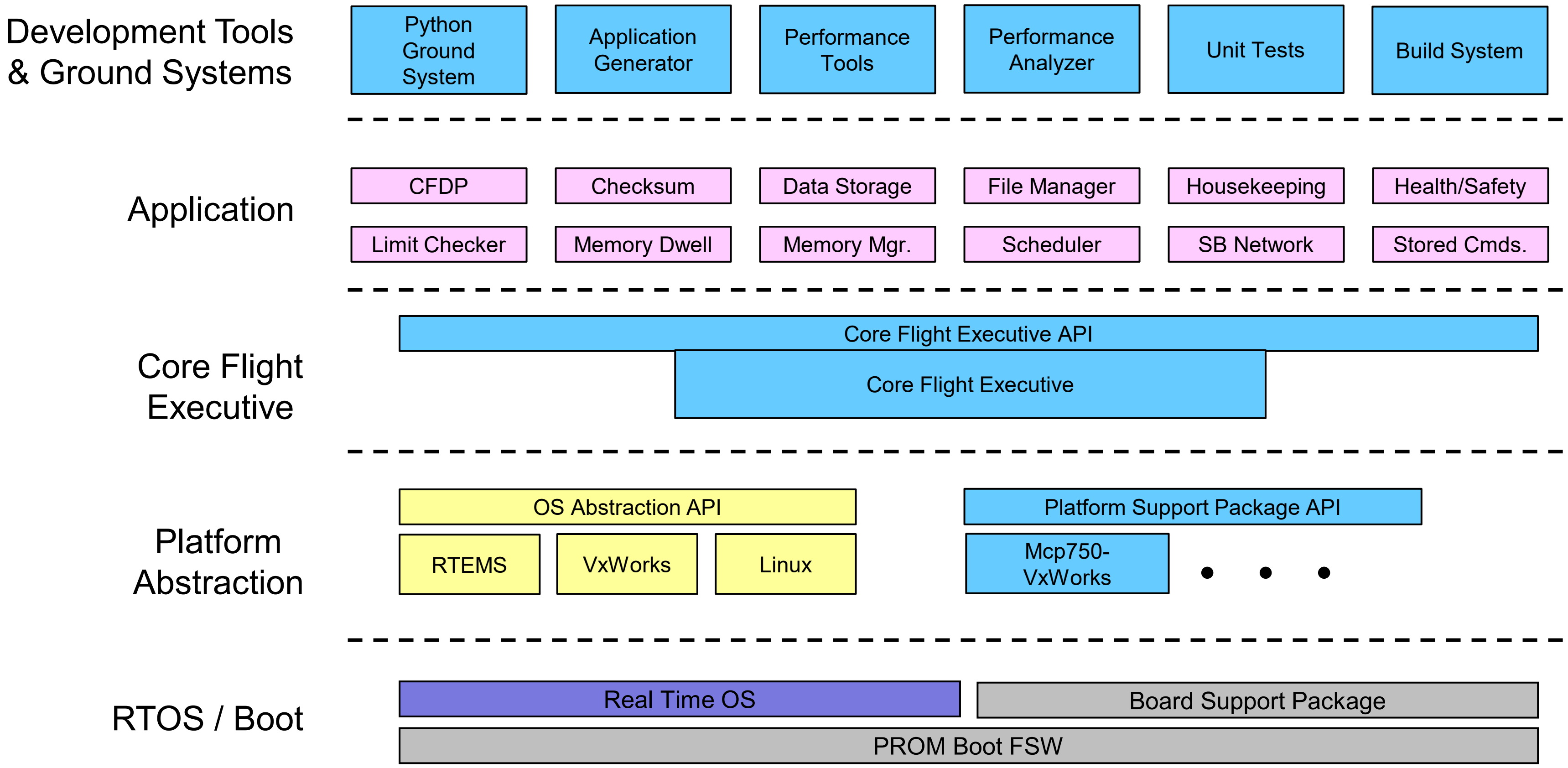
- **Feature Deprecation**
  - Mark feature as deprecated on any release
  - Provide tools/process that will warn applications when a feature is marked as deprecated
  - Only deprecate on major versions

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 & Ground Systems

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

Application

- CFDP
- Checksum
- Data Storage
- File Manager
- Housekeeping
- Health/Safety
- Limit Checker
- Memory Dwell
- Memory Mgr.
- Scheduler
- SB Network
- Stored Cmds.

Core Flight Executive

- Core Flight Executive API
- Core Flight Executive

Platform Abstraction

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

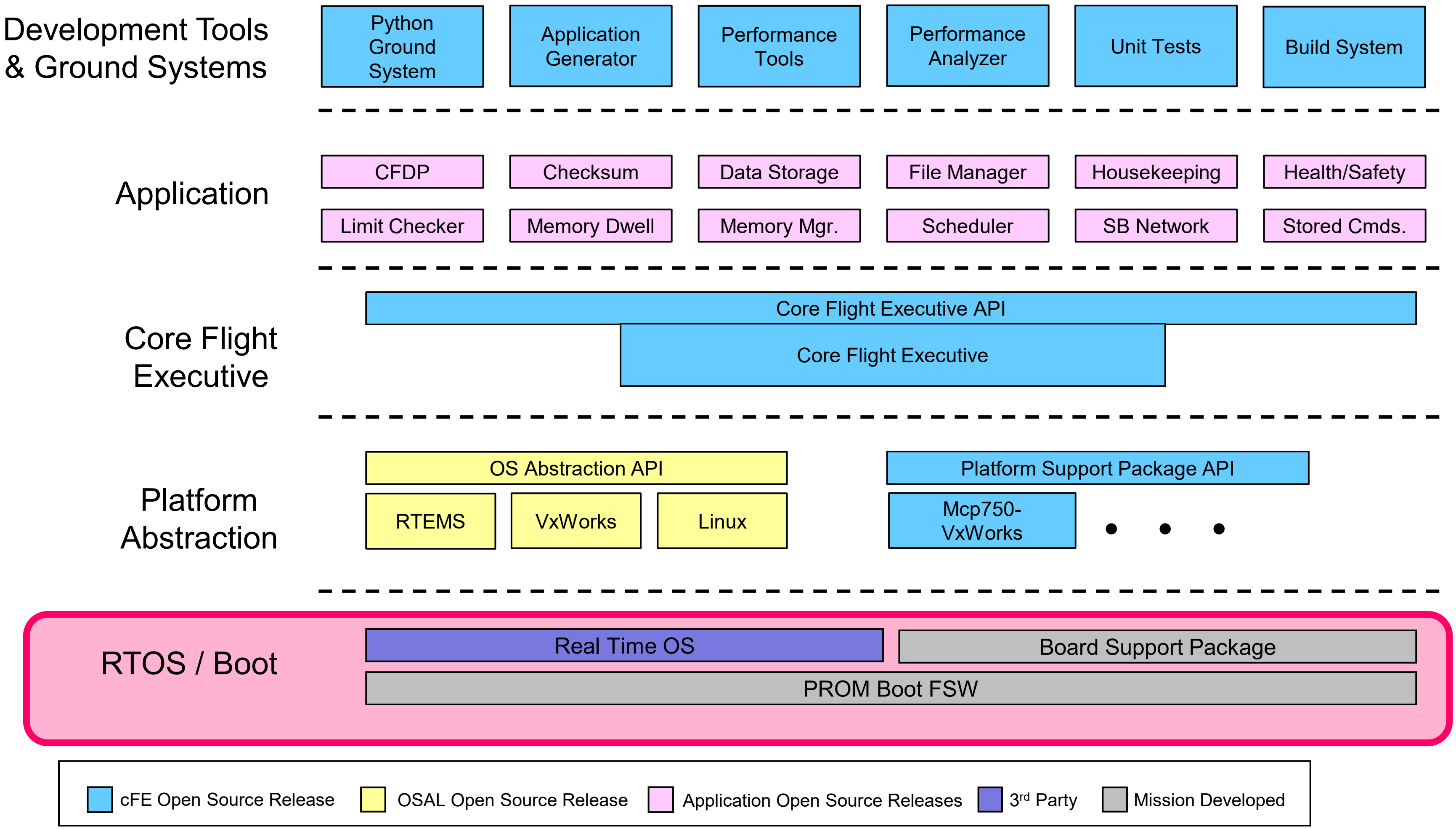
RTOS / Boot

- 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

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.



Development Tools & Ground Systems

Python Ground System

Application Generator

Performance Tools

Performance Analyzer

Unit Tests

Build System

Application

CFDP

Checksum

Data Storage

File Manager

Housekeeping

Health/Safety

Limit Checker

Memory Dwell

Memory Mgr.

Scheduler

SB Network

Stored Cmds.

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

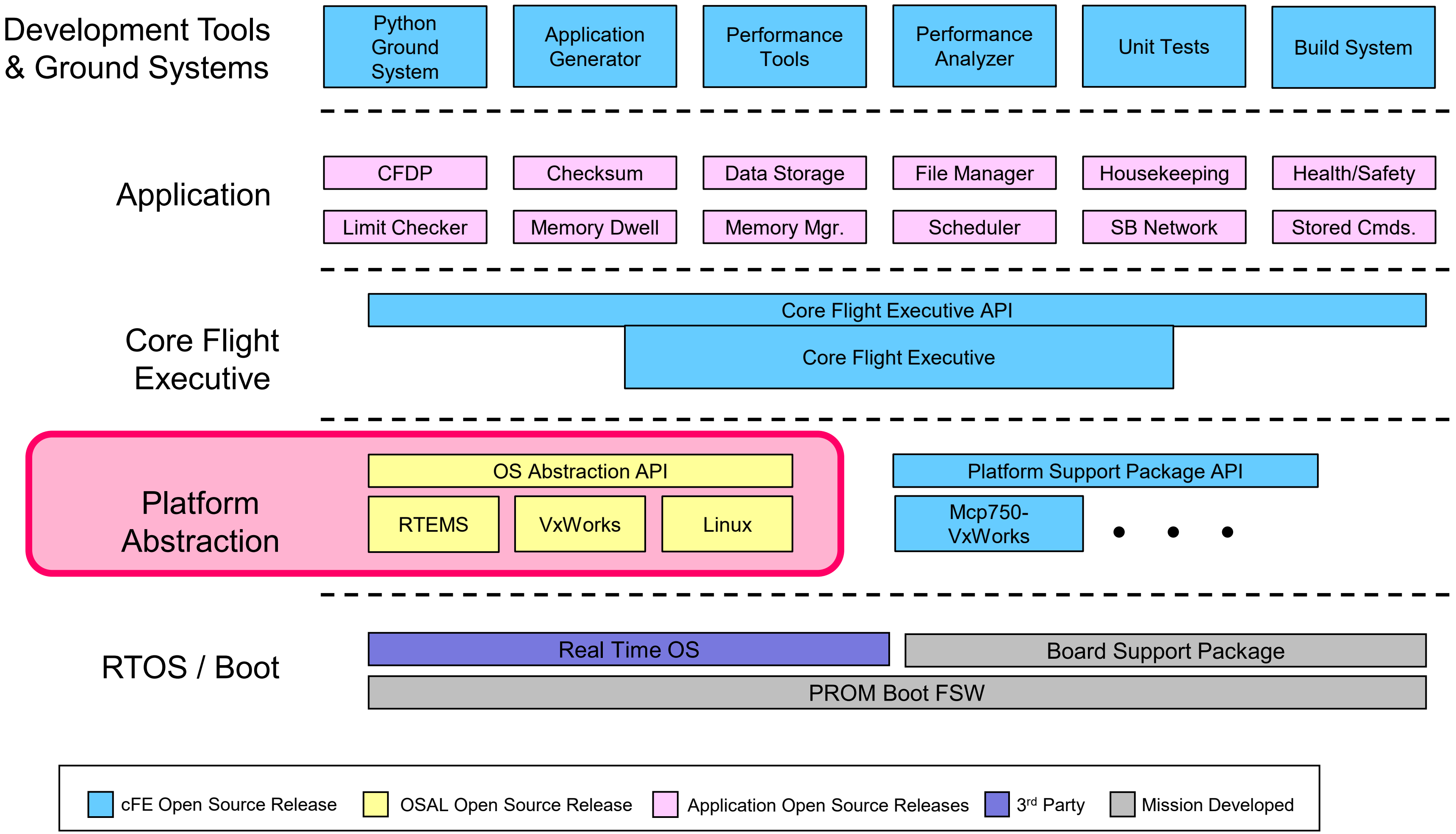
Board Support Package

PROM Boot FSW

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

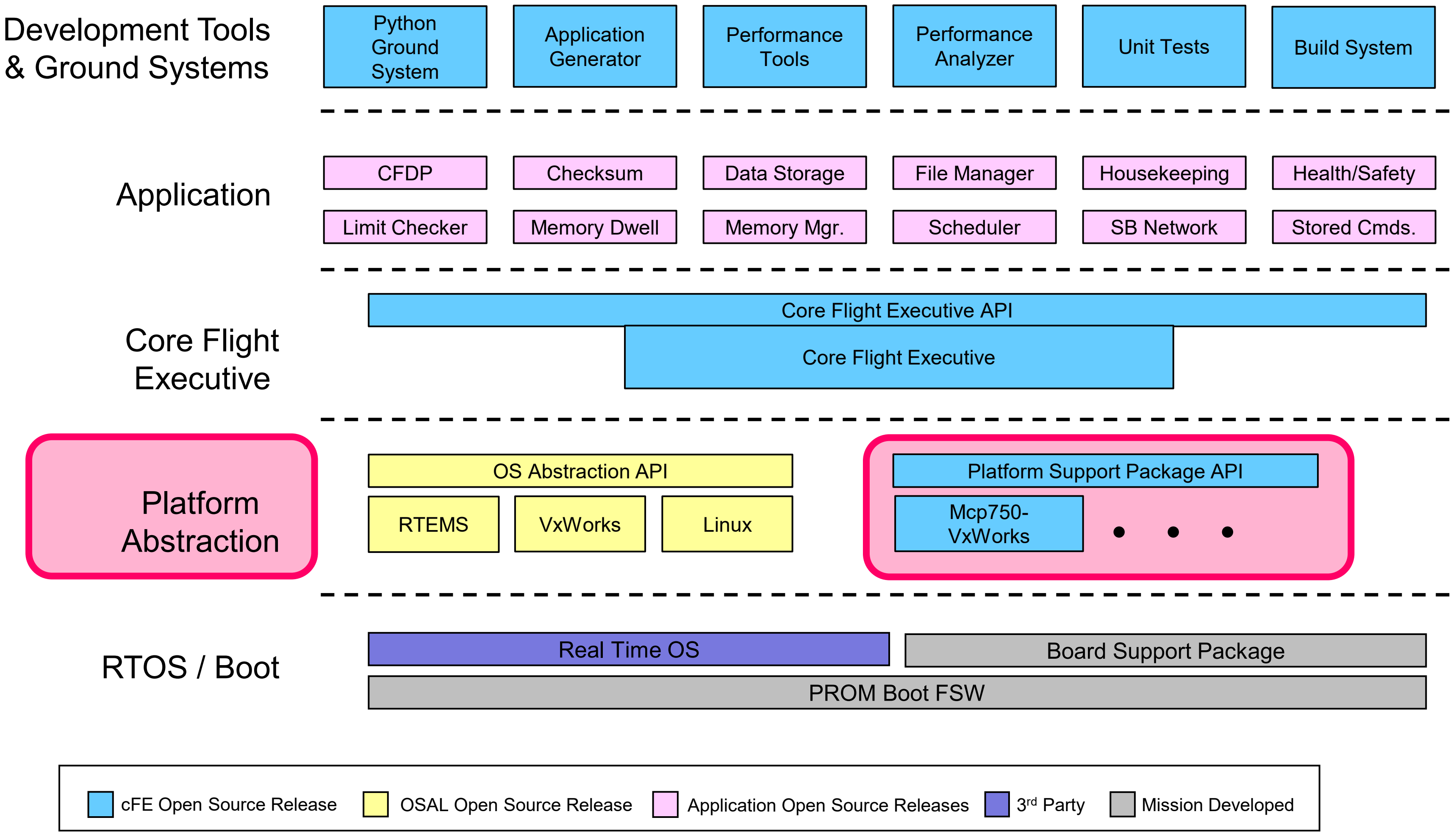
Platform Abstraction - OSAL

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.



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 & Ground Systems

Python Ground System

Application Generator

Performance Tools

Performance Analyzer

Unit Tests

Build System

Application

CFDP

Checksum

Data Storage

File Manager

Housekeeping

Health/Safety

Limit Checker

Memory Dwell

Memory Mgr.

Scheduler

SB Network

Stored Cmds.

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

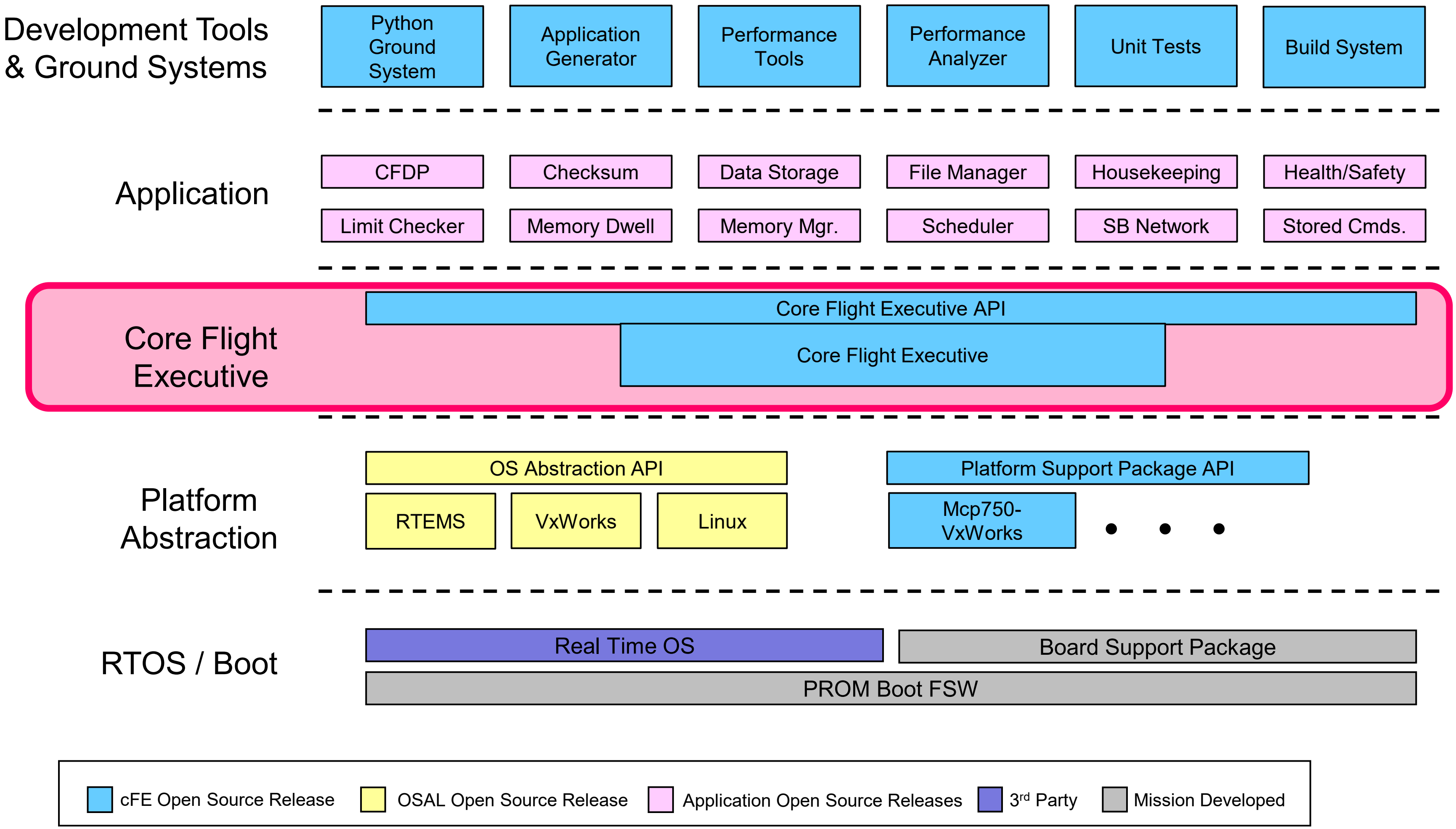
Board Support Package

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 & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

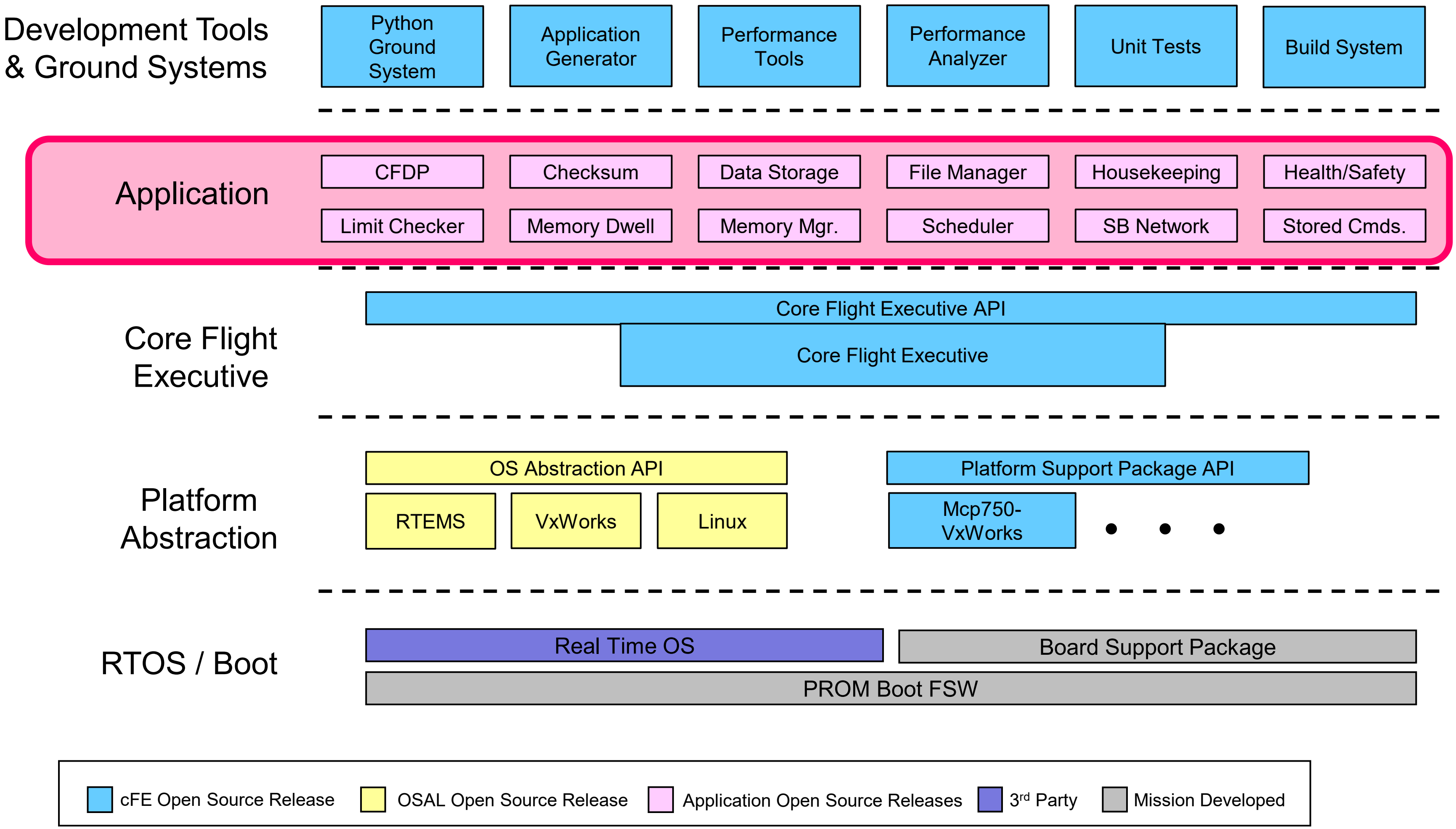
Real Time OS | Board Support Package

PROM Boot FSW

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

Applications

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



Development Tools & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

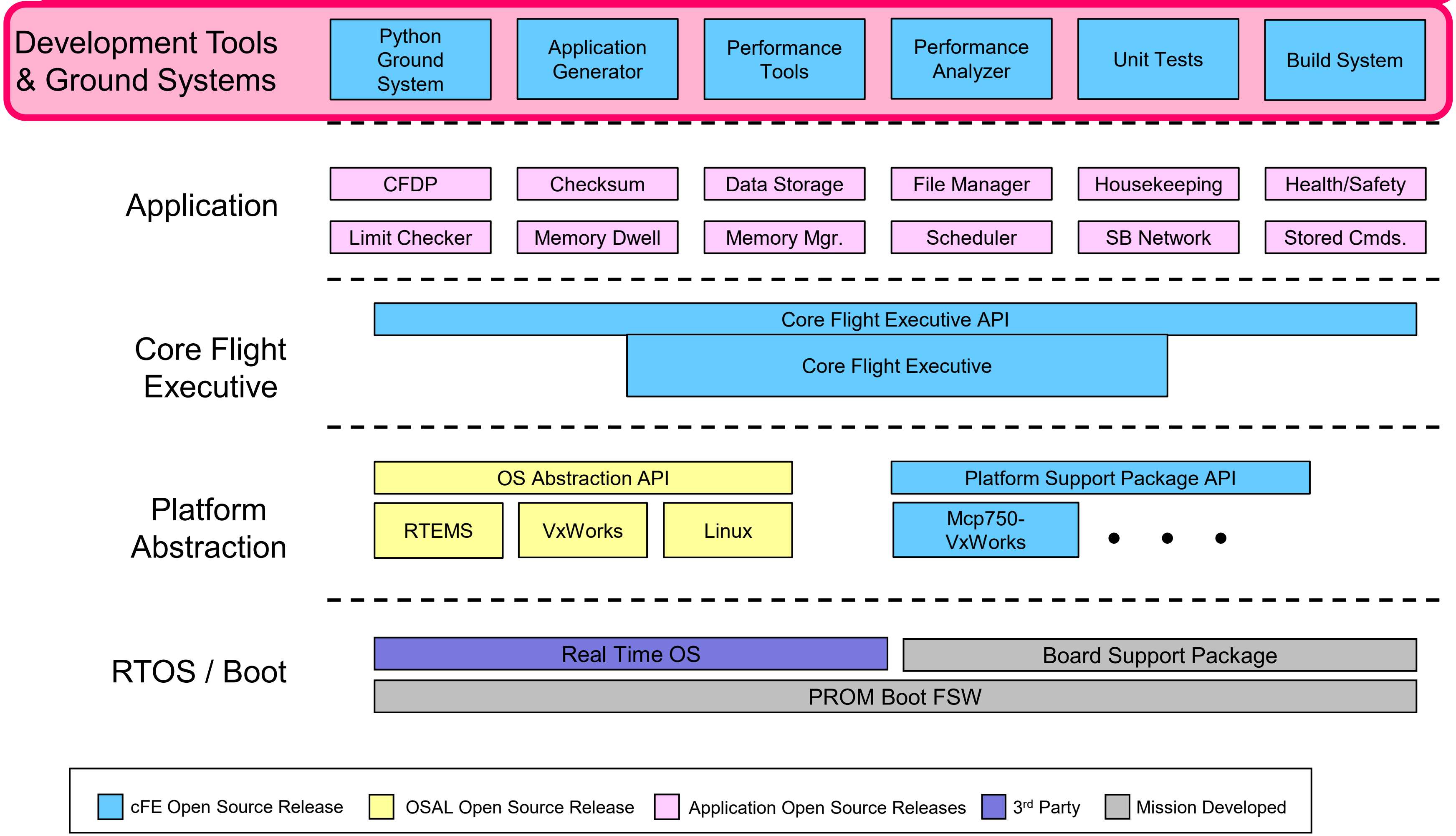
Real Time OS | Board Support Package

PROM Boot FSW

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

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 & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

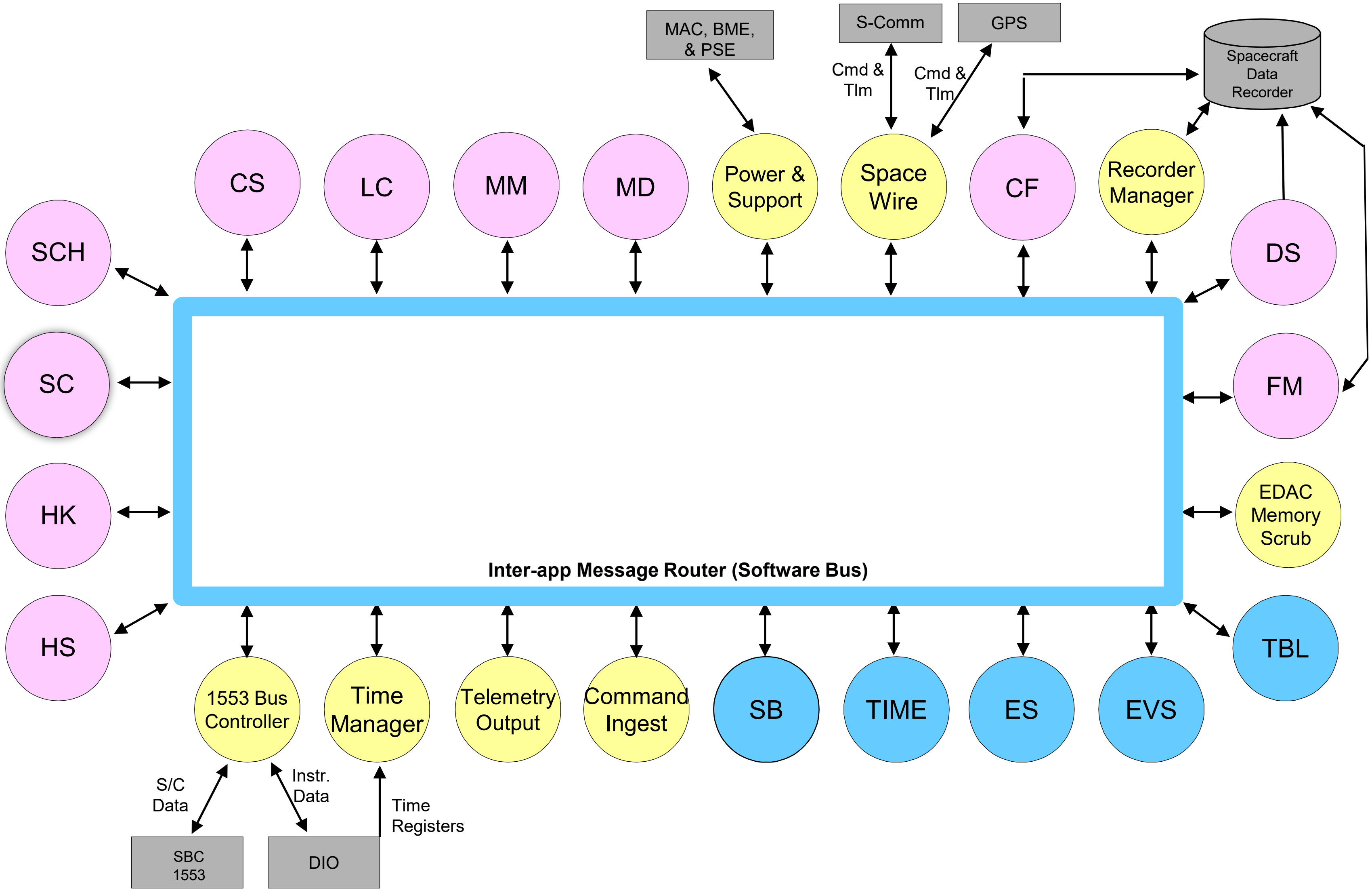
PROM Boot FSW

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

cFS Applications

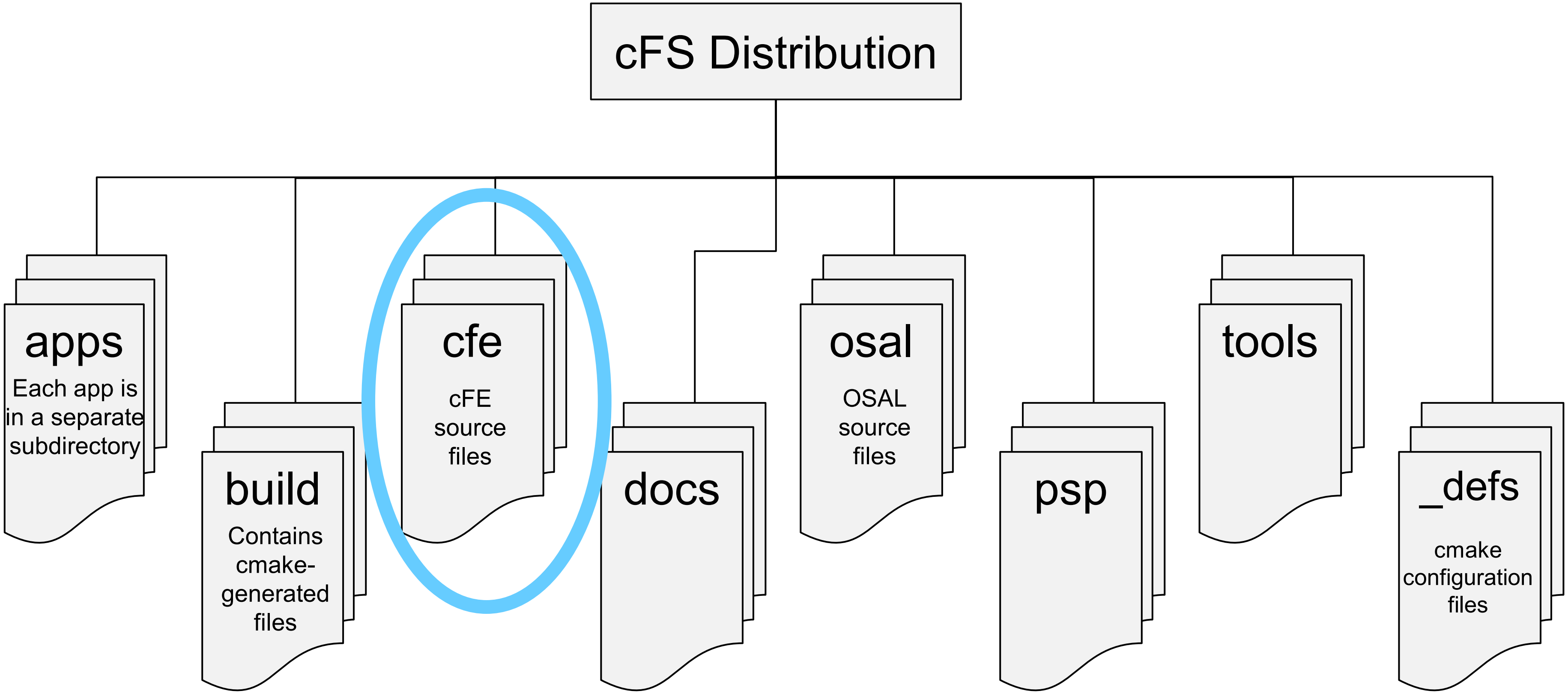
- **Can run anywhere the cFS framework has been deployed**
- **GSFC has released 12 applications that provide common command and data handling functionality such as**
  - Stored command management and execution
  - Onboard data storage file management
- **Missions use a combination of custom and reused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

Mission Application Example



●	CFS App
●	cFE Service
●	Mission C&DH App
●	Hardware

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

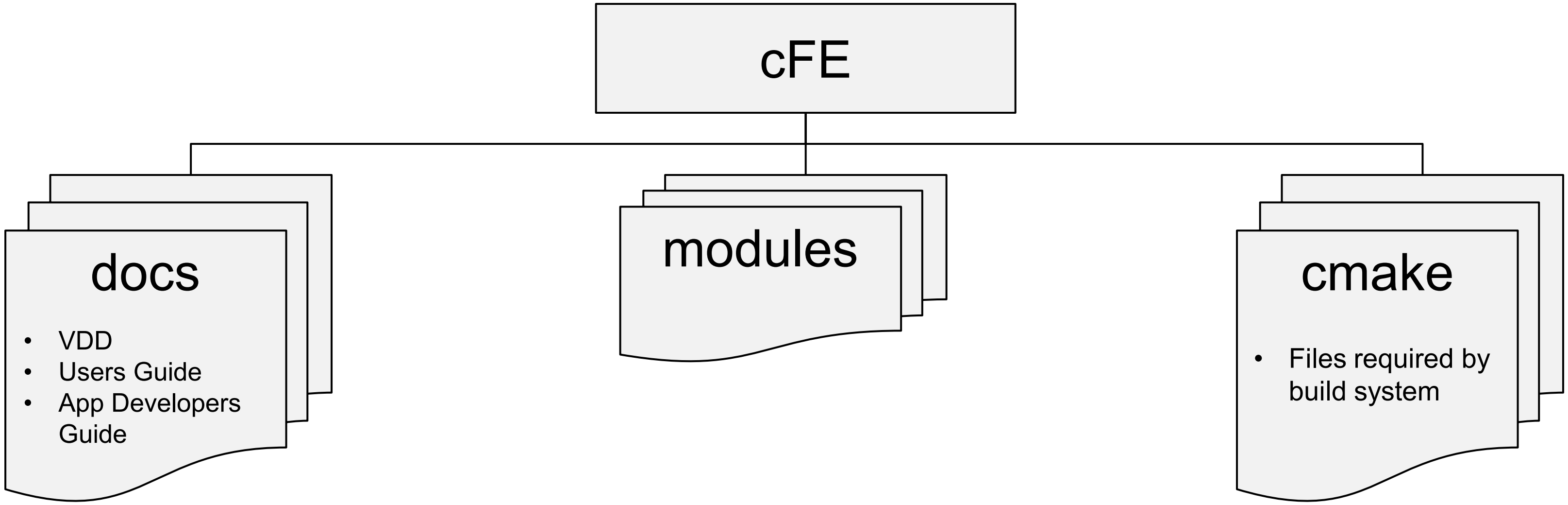
psp

tools

_defs

cmake configuration files

cFE Directory Structure



cFE

docs

- VDD
- Users Guide
- App Developers Guide

modules

cmake

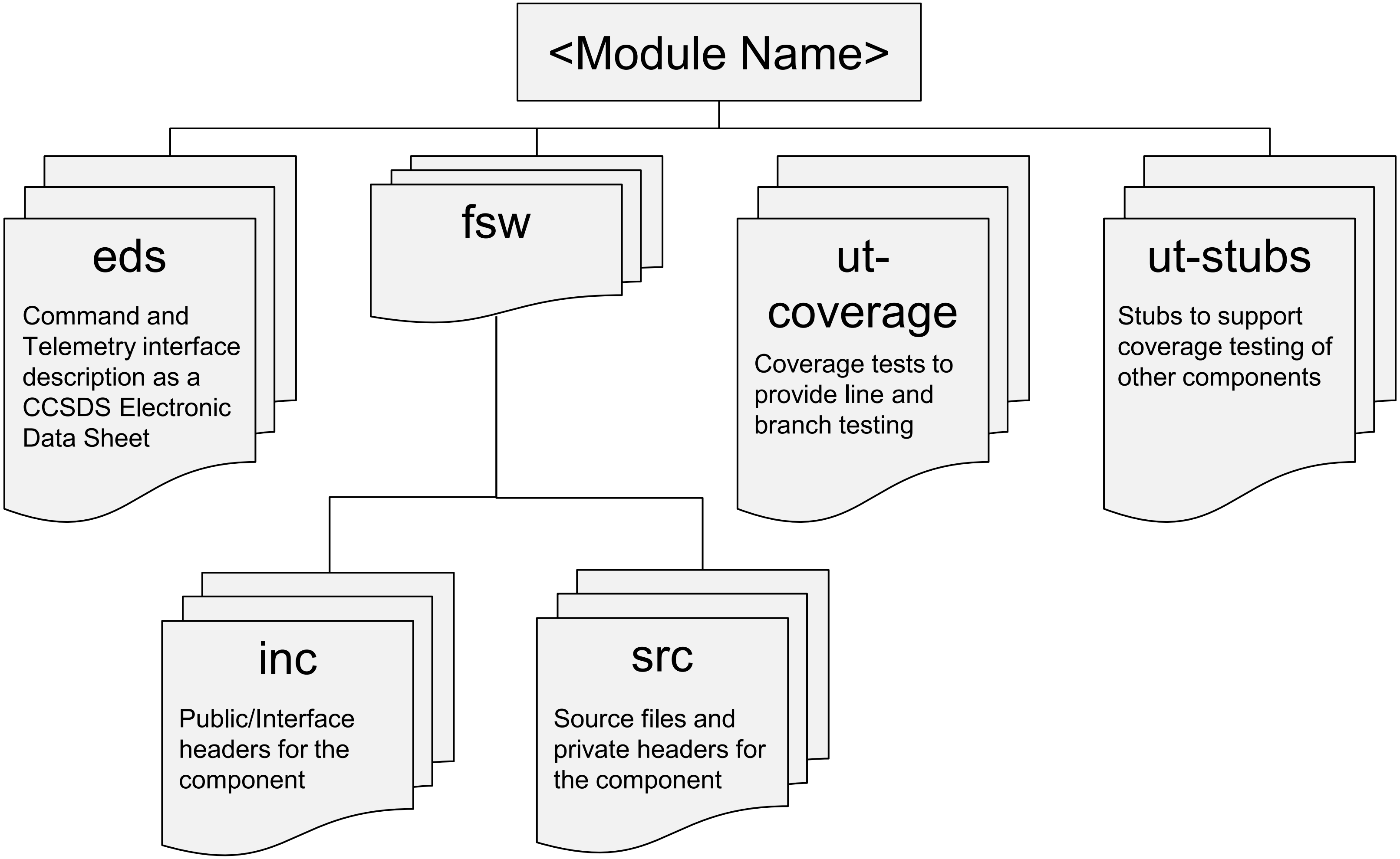
- Files required by build system

cFE Module Structure

- cFE core components are organized as modules
- Modular structure allows advanced users to add, remove, or override entire core services as necessary to support their particular mission requirements
- cFE “out of the box” provides reference implementations that meet the needs of most missions

New in Caelum

Module Directory Structure



<Module Name>

eds

Command and Telemetry interface description as a CCSDS Electronic Data Sheet

fsw

ut-coverage

Coverage tests to provide line and branch testing

ut-stubs

Stubs to support coverage testing of other components

inc

Public/Interface headers for the component

src

Source files and private headers for the component

Current Modules

Module	Purpose/Content
cfe_assert	A CFE-compatible library wrapping the basic UT assert library.
cfe_testcase	A CFE-compatible library implementing test cases for CFE core apps.
core_api	Contains the public interface definition of the complete CFE core - public API/headers only, no implementation.
core_private	Contains the inter-module interface definition of the CFE core - internal API/headers only, no implementation.
es	Implementation of the Executive Services (ES) core module.
evs	Implementation of the Event Services (EVS) core module.
fs	Implementation of the File Services (FS) core module.
msg	Implementation of the Message (MSG) core module.
resourceid	Implementation of the Resource ID core module.
sb	Implementation of the Software Bus (SB) core module.
sbr	Implementation of the Software Bus (SB) Routing module.
tbl	Implementation of the Table Services (TBL) core module.
time	Implementation of the Time Services (TIME) core module.

Module 1: Backup Charts

cFS References

Where is the cFS?

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

- **OSAL, https://github.com/nasa/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

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

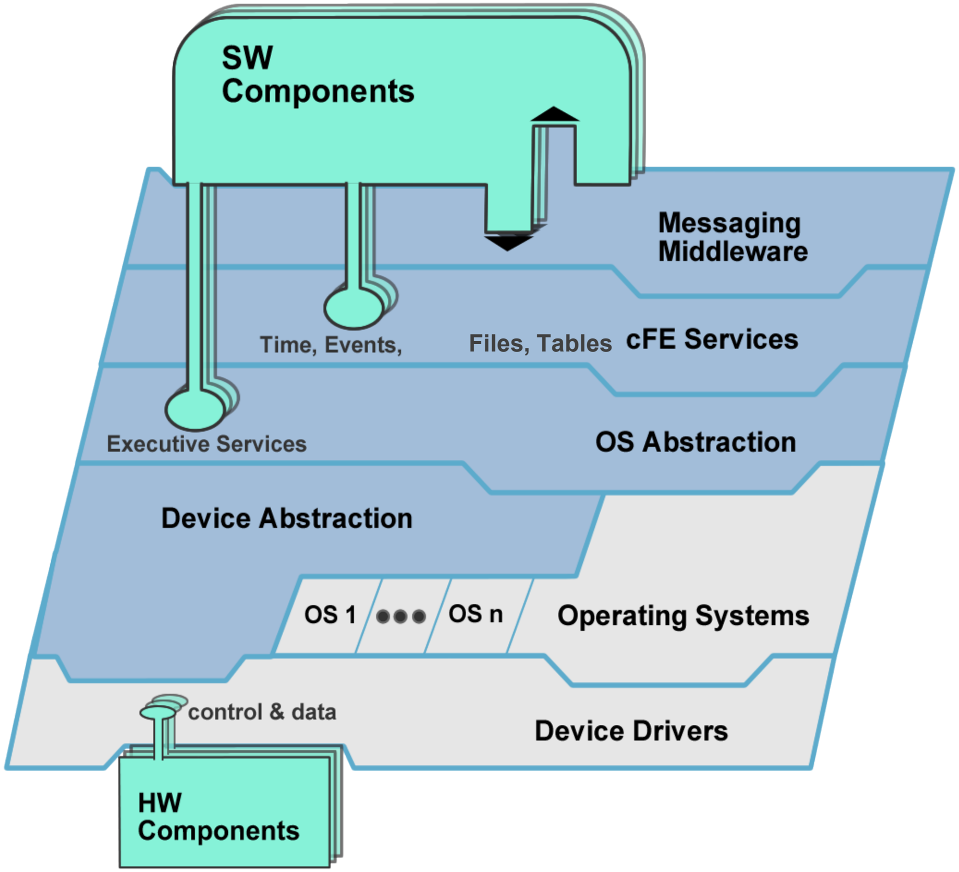
- **Effective Reuse**
  - The architecture must support an effective reuse approach. This includes the software and artifacts (e.g. requirements, design, code, review presentations, tests, operations guides, command and telemetry databases). The goal is to achieve 100% reuse of a software component with 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
- **Predictable 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.



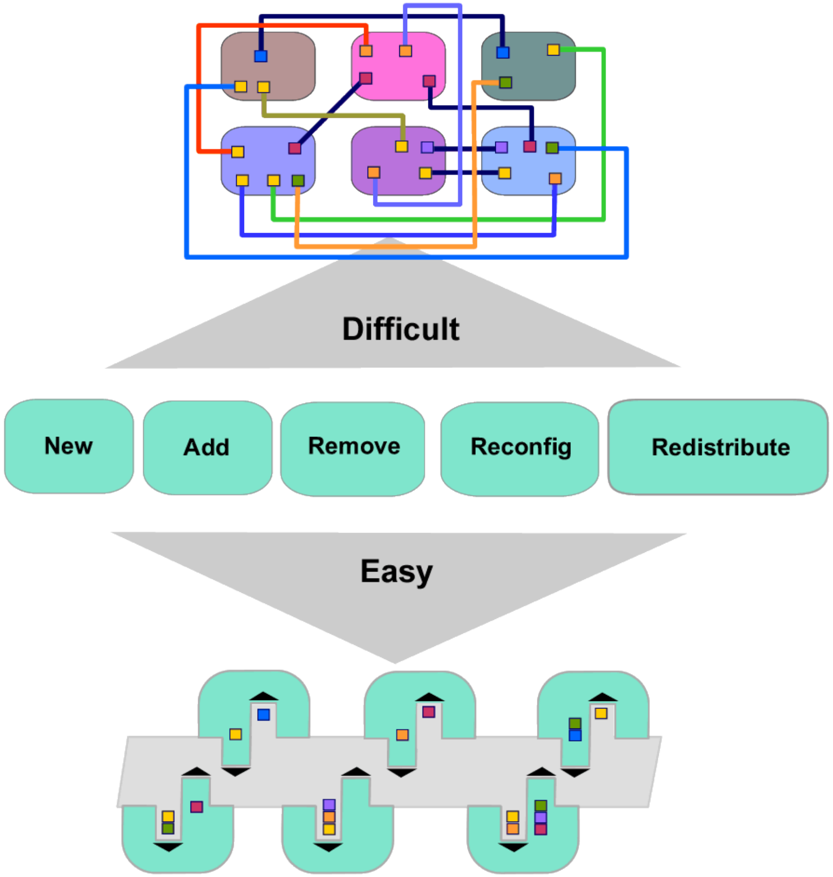
Plug and Play

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.

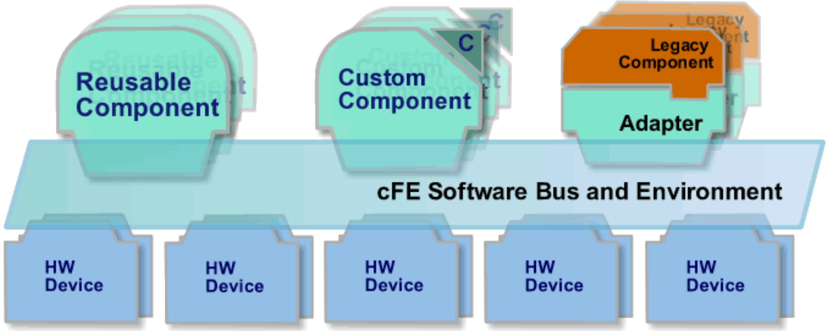
Reusable Components

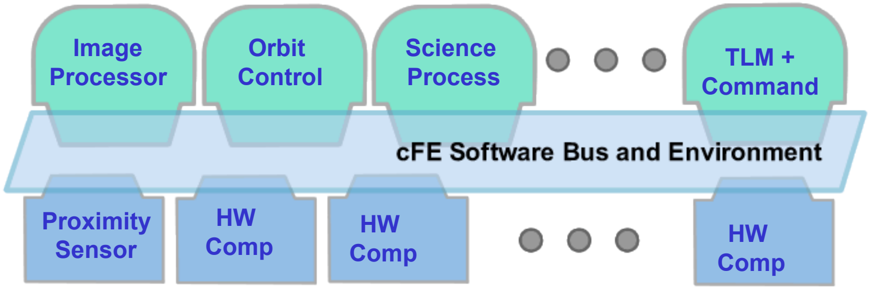
Reusable Components

- 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".





National Aeronautics and Space Administration

Core Flight System (cFS) Training

Module 2: Core Flight Executive (cFE)

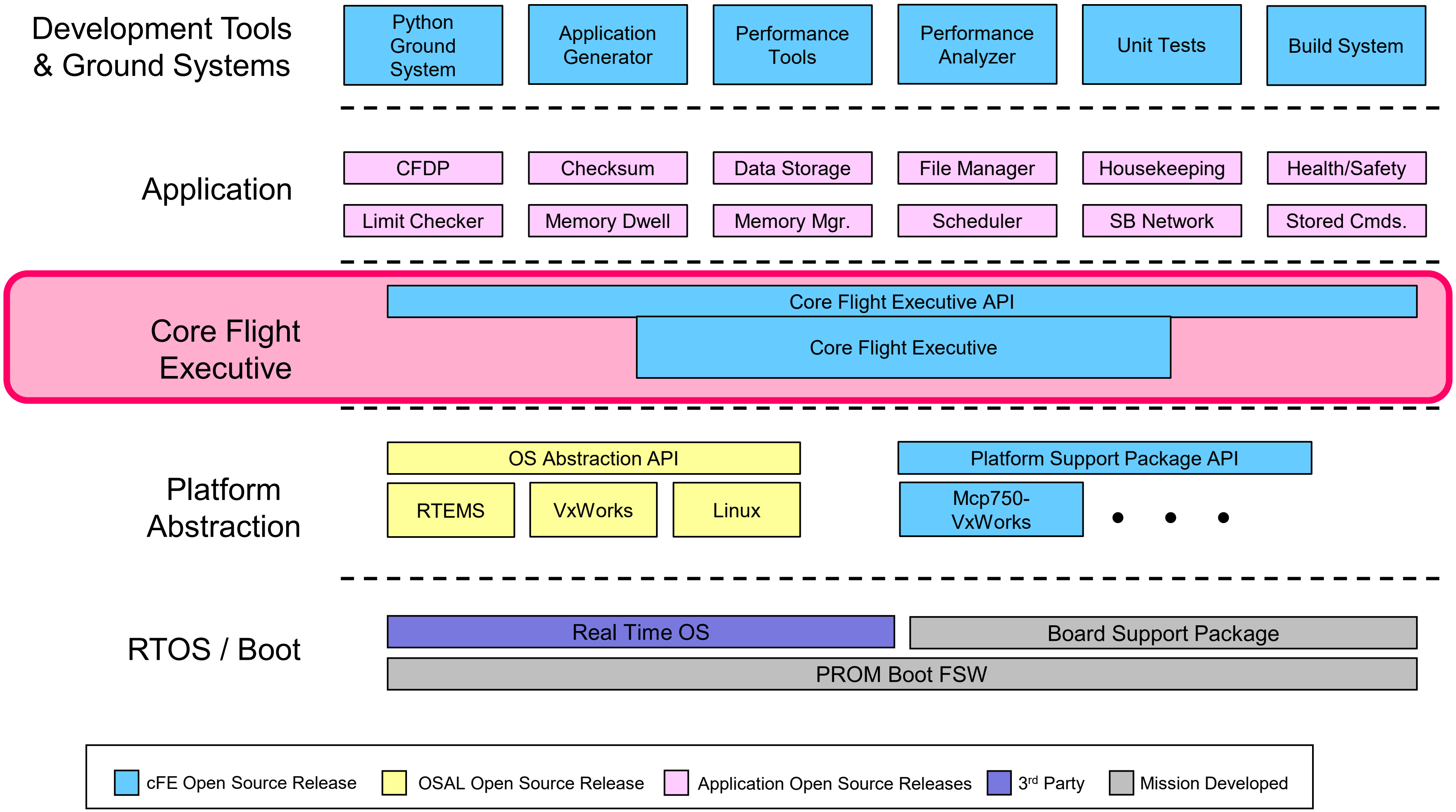
Services

August 3, 2019

Course Agenda

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

cFE Services - cFS Context



Development Tools & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

What are the cFE Services?

Executive Services (ES)

- Manages the software system and creates an application runtime environment

Software Bus (SB) Services

- Provides an application publish/subscribe messaging service

Event Services (EVS)

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

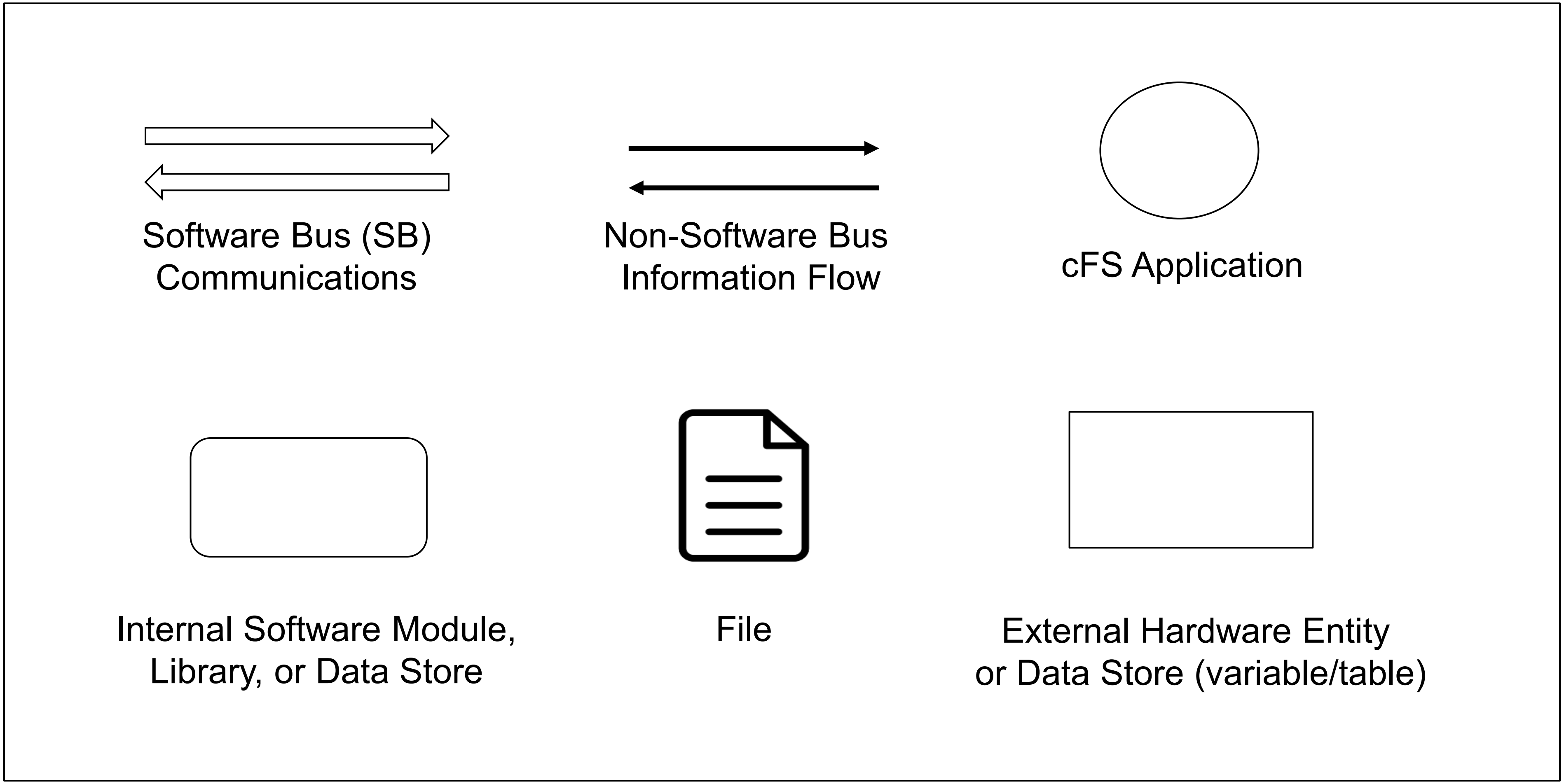
Time Services (TIME)

- Manages spacecraft time

Table Services (TBL)

- Manages application table images

Diagram Notation



Software Bus (SB) Communications

Non-Software Bus Information Flow

cFS Application

Internal Software Module, Library, or Data Store

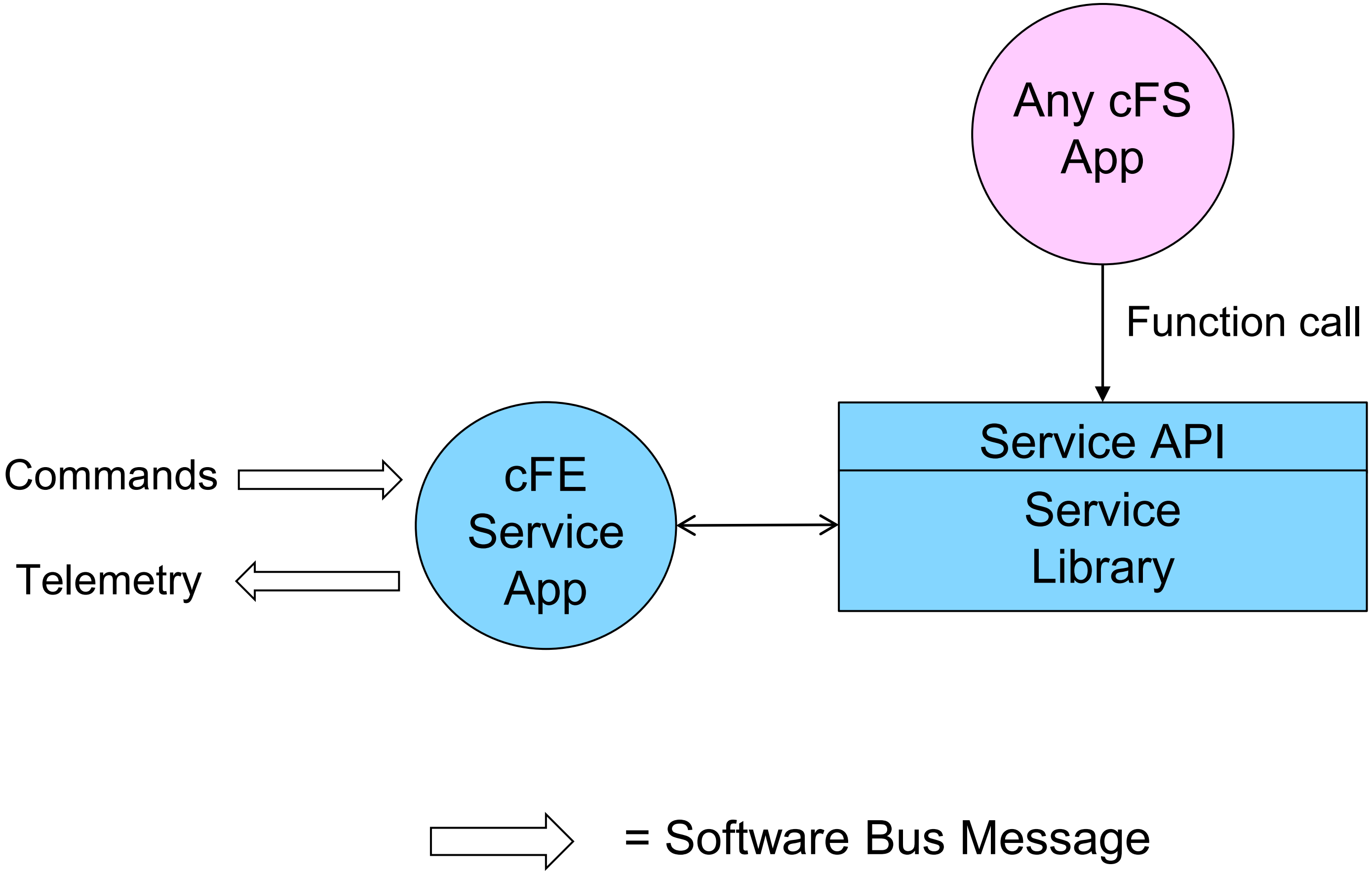
File

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 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.)**

- **Platform Configuration parameters – used for the specific processor (e.g. time client/server config, max number of applications, max number of tables, etc.)**

- **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)
default_osconfig.cmake	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 checkout caelum-rc3
git submodule update --init
```

Subsequent exercises assume that cFS was cloned into the home directory (“~/cFS”)

Copy in the default makefile and definitions:

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

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:

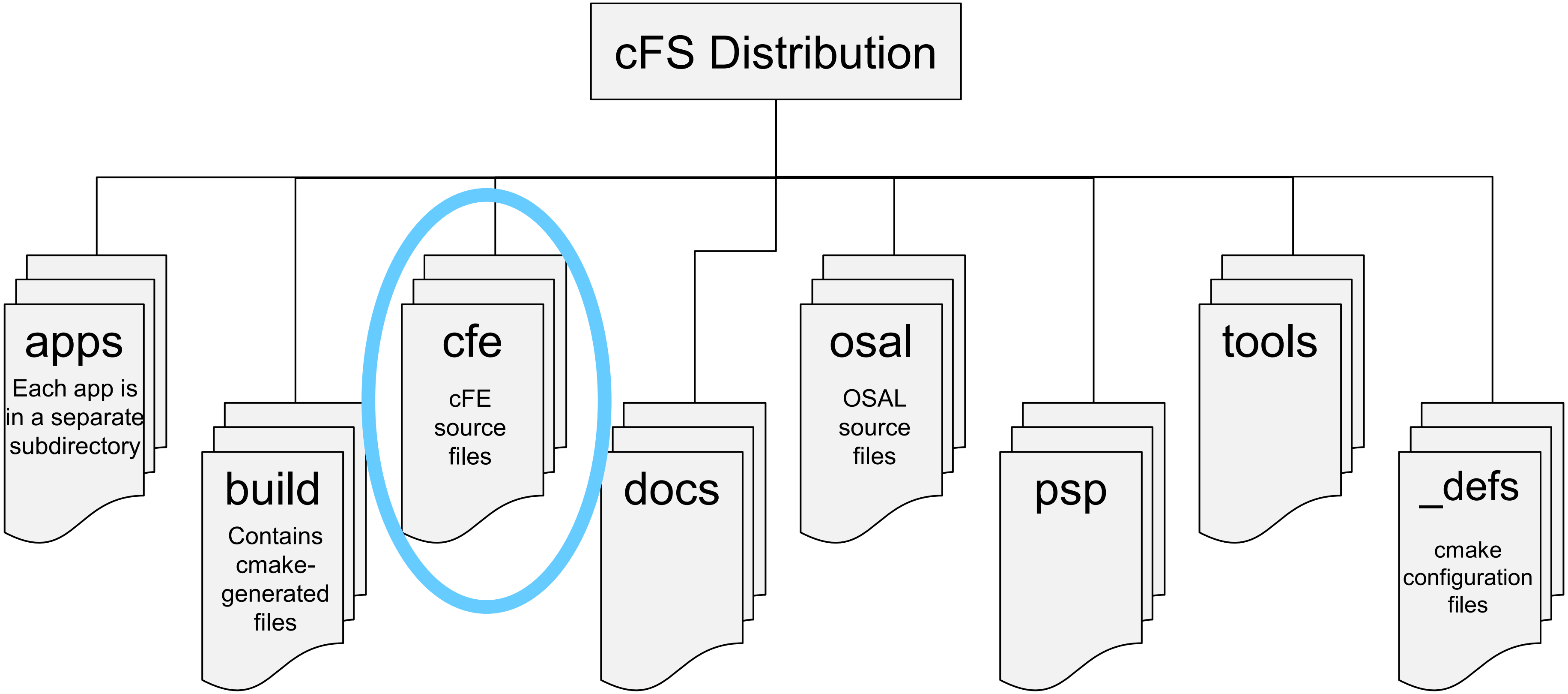
To prep, compile, and run (from cFS directory above):

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

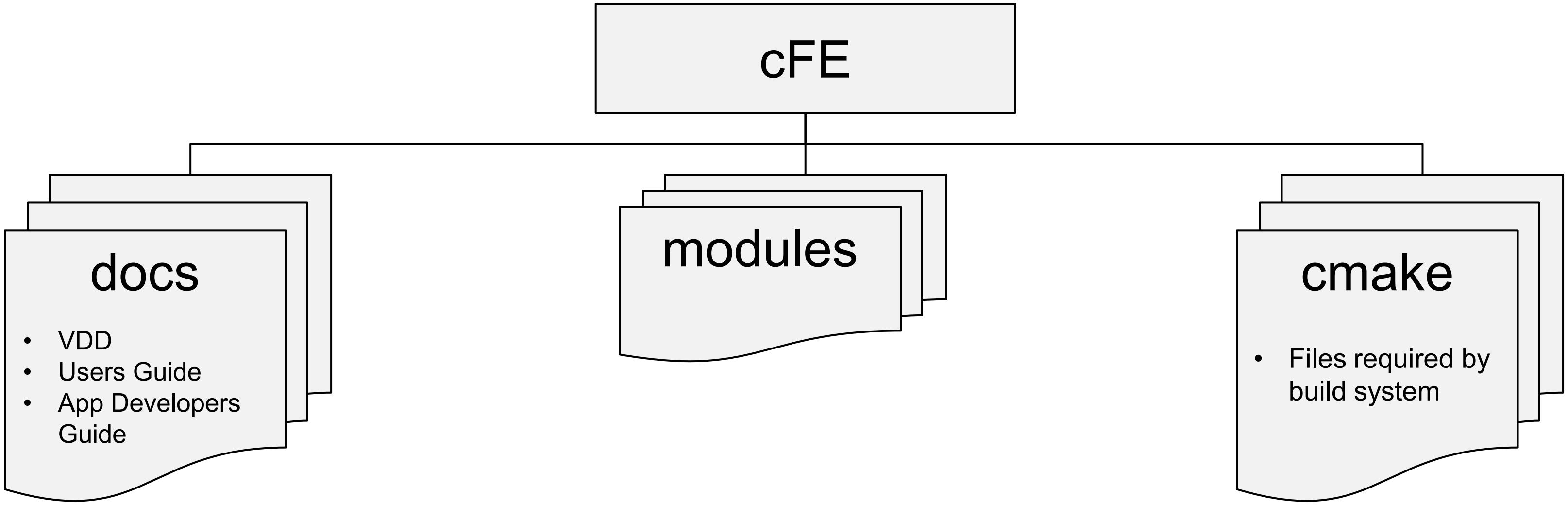
Shortcut:
"`make SIMULATION=native install`" will do the prep/make/install steps in one call.

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 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

docs

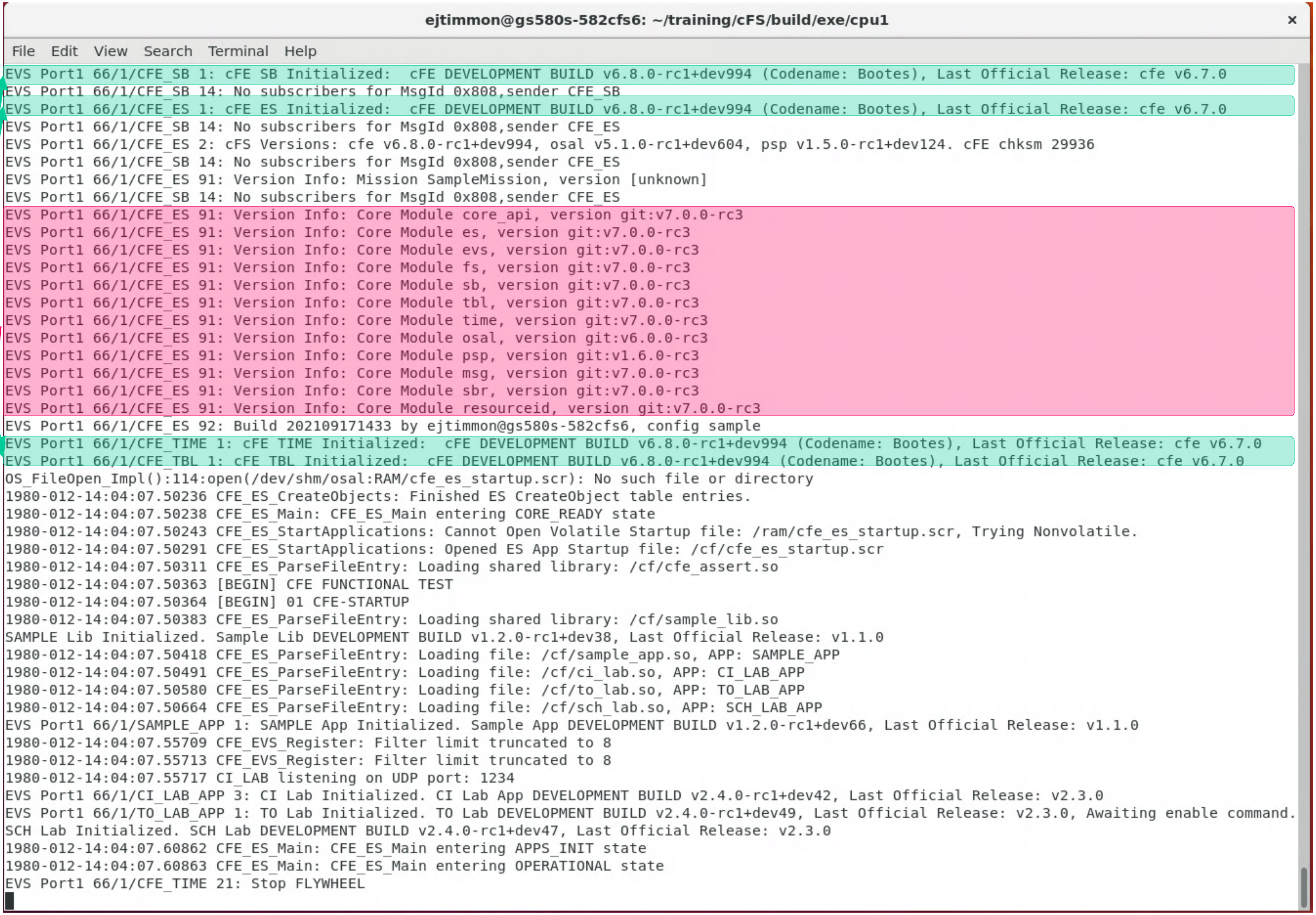
- VDD
- Users Guide
- App Developers Guide

modules

cmake

- Files required by build system

Exercise 1 Recap



```
EVS Port1 66/1/CFE_SB 1: cFE SB Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_SB
EVS Port1 66/1/CFE_ES 1: cFE ES Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 66/1/CFE_ES 2: cFS Versions: cfe v6.8.0-rc1+dev994, osal v5.1.0-rc1+dev604, psp v1.5.0-rc1+dev124. cFE chksm 29936
EVS Port1 66/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 66/1/CFE_ES 91: Version Info: Mission SampleMission, version [unknown]
EVS Port1 66/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module core_api, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module es, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module evs, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module fs, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sb, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module tbl, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module time, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module osal, version git:v6.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module psp, version git:v1.6.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module msg, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sbr, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module resourceid, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 92: Build 202109171433 by ejtimmon@gs580s-582cfs6, config sample
EVS Port1 66/1/CFE_TIME 1: cFE TIME Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 1: cFE TBL Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
OS_FileOpen_Impl():114:open(/dev/shm/osal:RAM/cfe_es_startup.scr): No such file or directory
1980-012-14:04:07.50236 CFE_ES_CreateObjects: Finished ES CreateObject table entries.
1980-012-14:04:07.50238 CFE_ES_Main: CFE_ES_Main entering CORE_READY state
1980-012-14:04:07.50243 CFE_ES_StartApplications: Cannot Open Volatile Startup file: /ram/cfe_es_startup.scr, Trying Nonvolatile.
1980-012-14:04:07.50291 CFE_ES_StartApplications: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:04:07.50311 CFE_ES_ParseFileEntry: Loading shared library: /cf/cfe_assert.so
1980-012-14:04:07.50363 [BEGIN] CFE FUNCTIONAL TEST
1980-012-14:04:07.50364 [BEGIN] 01 CFE-STARTUP
1980-012-14:04:07.50383 CFE_ES_ParseFileEntry: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.2.0-rc1+dev38, Last Official Release: v1.1.0
1980-012-14:04:07.50418 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:04:07.50491 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:04:07.50580 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:04:07.50664 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-14:04:07.55709 CFE_EVS_Register: Filter limit truncated to 8
1980-012-14:04:07.55713 CFE_EVS_Register: Filter limit truncated to 8
1980-012-14:04:07.55717 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.4.0-rc1+dev42, Last Official Release: v2.3.0
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable command.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
1980-012-14:04:07.60862 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-14:04:07.60863 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
```

cFE Services Initialized

Version info for each module

National Aeronautics and Space Administration

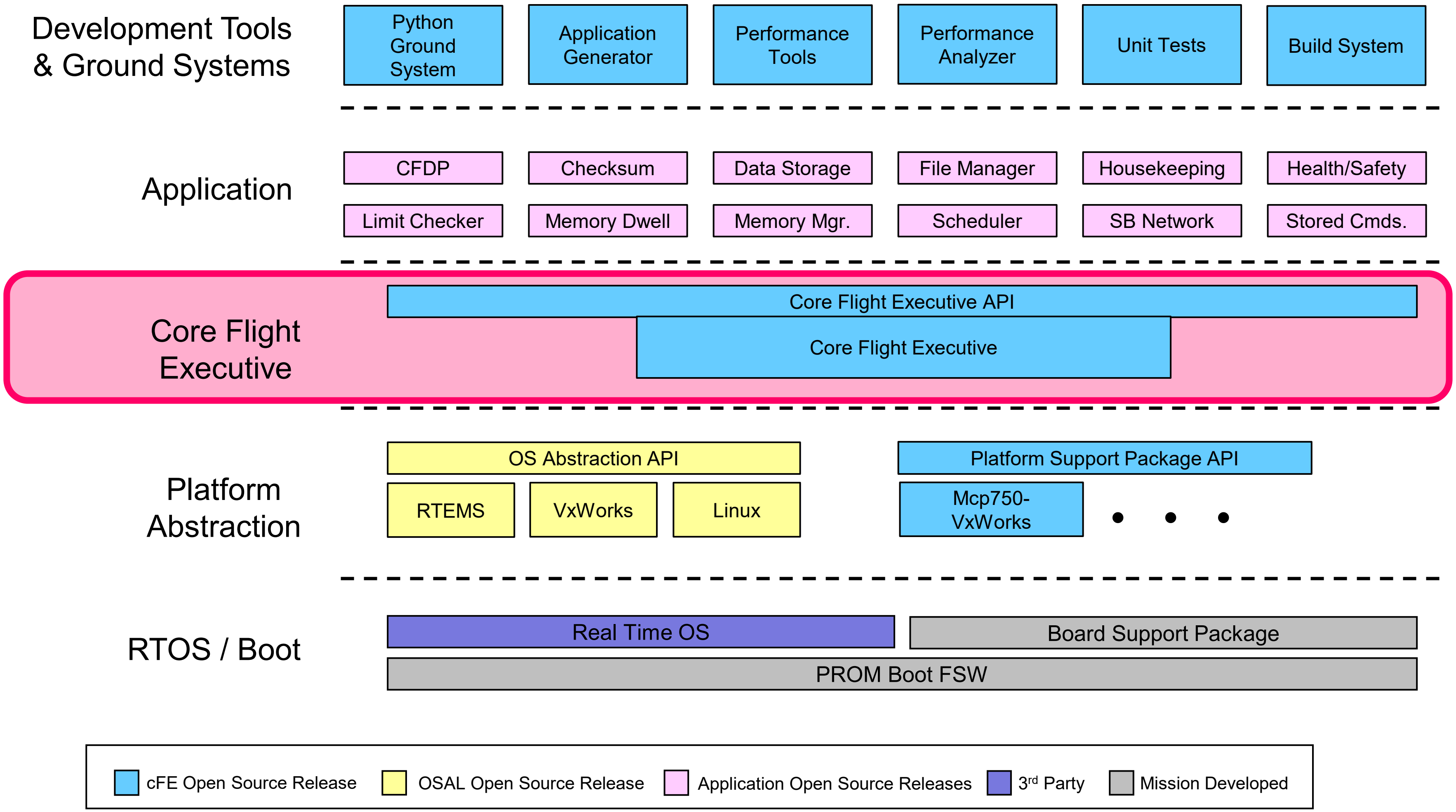
Core Flight System (cFS) Training

Module 2a: Executive Services

Course Agenda

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

Executive Services - cFS Context



Development Tools & Ground Systems

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

Application

- CFDP
- Checksum
- Data Storage
- File Manager
- Housekeeping
- Health/Safety
- Limit Checker
- Memory Dwell
- Memory Mgr.
- Scheduler
- SB Network
- Stored Cmds.

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

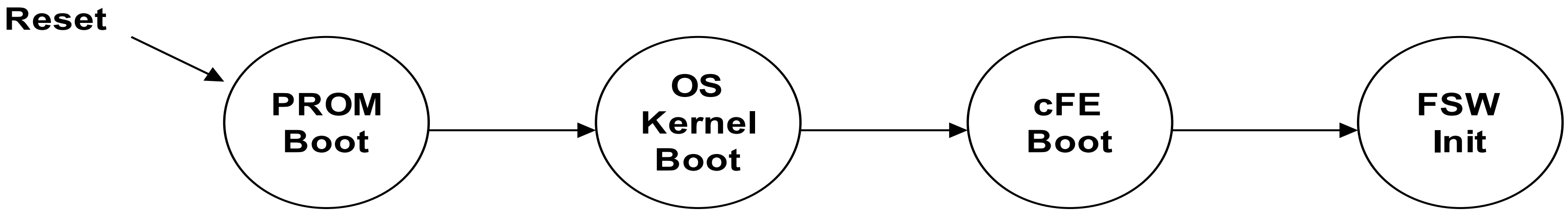
□ cFE Open Source Release □ OSAL Open Source Release □ Application Open Source Releases □ 3rd Party □ Mission Developed

Executive Services (ES) – Overview

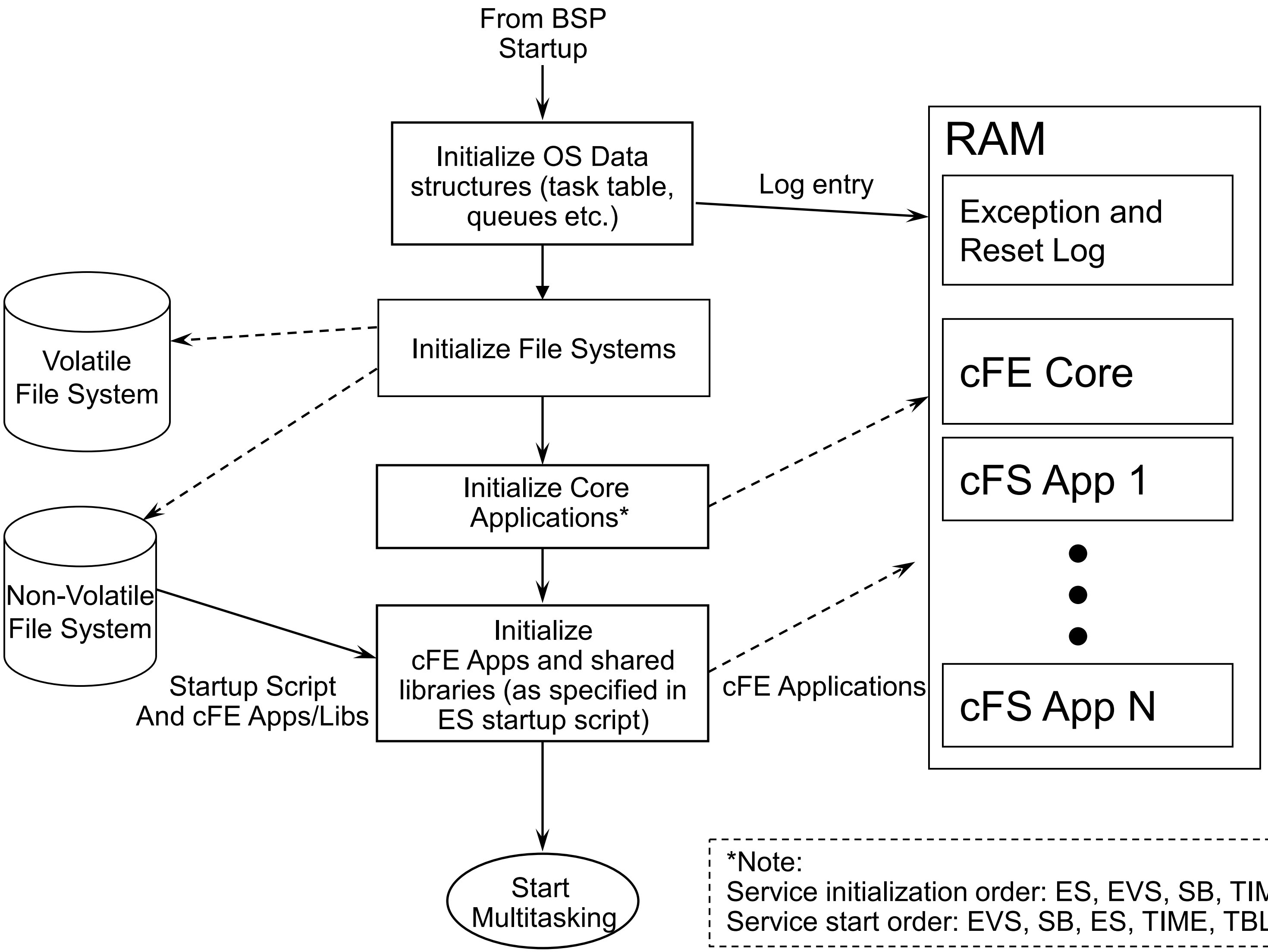
- **Initializes the cFE**
  - Reports reset type
  - Maintains an exception-reset log across processor resets
- **Creates the application runtime environment**
  - Primary interface to underlying operating system task services
  - Manages application resources
  - Starts initial applications according 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 services (ES, EVS, TBL, SB, & TIME)
  - Starts the 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
cFE Core
Initialize Core Applications*
cFS App 1
Non-Volatile File System
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

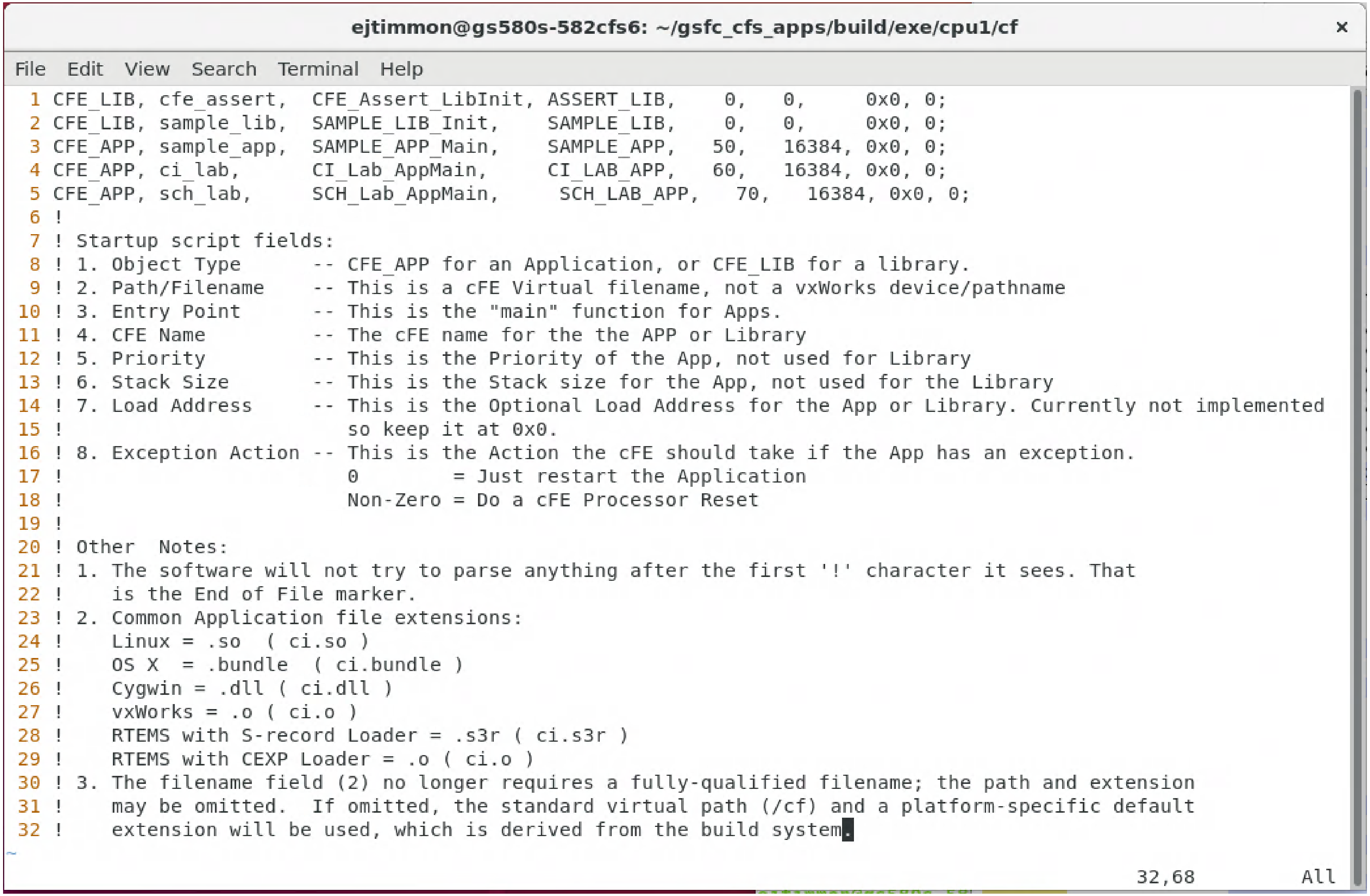
The cFE core is started as one unit. The cFE Core is linked with the RTOS and support libraries and loaded into system EEPROM as a static executable.

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_LIB, cfe_assert,  CFE_Assert_LibInit, ASSERT_LIB,    0,    0,      0x0, 0;
CFE_LIB, sample_lib,  SAMPLE_LIB_Init,    SAMPLE_LIB,    0,    0,      0x0, 0;
CFE_APP, sample_app,  SAMPLE_APP_Main,    SAMPLE_APP,   50,   16384, 0x0, 0;
CFE_APP, ci_lab,      CI_Lab_AppMain,     CI_LAB_APP,   60,   16384, 0x0, 0;
CFE_APP, sch_lab,     SCH_Lab_AppMain,     SCH_LAB_APP,   70,   16384, 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 )
! 3. The filename field (2) no longer requires a fully-qualified filename; the path and extension
!    may be omitted.  If omitted, the standard virtual path (/cf) and a platform-specific default
!    extension will be used, which is derived from the build system.
```

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

- **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 (1 of 6)

Resource ID APIs	Purpose
CFE_ES_AppID_ToIndex	Calculates a zero-based integer value that may be used for indexing into a local resource table/array.
CFE_ES_LibID_ToIndex	Calculates a zero-based integer value that may be used for indexing into a local resource table/array.
CFE_ES_TaskID_ToIndex	Calculates a zero-based integer value that may be used for indexing into a local resource table/array.
CFE_ES_CounterID_ToIndex	Calculates a zero-based integer value that may be used for indexing into a local resource table/array.
Entry/Exit APIs	**Purpose**
CFE_ES_Main	This is the entry point into the cFE software.
CFE_ES_ResetCFE	This API causes an immediate reset of the cFE Kernel and all cFE Applications.

Application Control APIs	Purpose
CFE_ES_RestartApp	This API causes a cFE Application to be unloaded and restarted from the same file as the last start.
CFE_ES_ReloadApp	This API causes a cFE Application to be stopped and restarted from the specified file.
CFE_ES_DeleteApp	This API causes a cFE Application to be stopped deleted.

Executive Services – APIs (2 of 6)

App Behavior APIs	Purpose
CFE_ES_ExitApp	This API is the "Exit Point" for the cFE application
CFE_ES_RunLoop	This is the API that allows an app to check for exit requests from the system, or request shutdown from the system.
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_IncrementTaskCounter	Increments the execution counter for the calling task

Child Task APIs	Purpose
CFE_ES_CreateChildTask	Creates a new task under an existing Application
CFE_ES_GetTaskIDByName	Get a Task ID associated with a specified Task name
CFE_ES_GetTaskName	Get a Task name for a specified Task ID
CFE_ES_DeleteChildTask	Deletes a task under an existing Application
CFE_ES_ExitChildTask	Exits a child task

Executive Services – APIs (3 of 6)

cFE Information APIs	Purpose
CFE_ES_GetResetType	Return the most recent Reset Type
CFE_ES_GetAppID	Get an Application ID for the calling Application
CFE_ES_GetTaskID	Get the task ID of the calling context
CFE_ES_GetAppIDByName	Get an Application ID associated with a specified Application name
CFE_ES_GetLibIDByName	Get a Library ID associated with a specified Library name
CFE_ES_GetAppName	Get an Application name for a specified Application ID
CFE_ES_GetLibName	Get a Library name for a specified Library ID
CFE_ES_GetAppInfo	Get Application Information given a specified App ID
CFE_ES_GetTaskInfo	Get Task Information given a specified Task ID
CFE_ES_GetLibInfo	Get Library Information given a specified Resource ID
CFE_ES_GetModuleInfo	Get Information given a specified Resource ID

Executive Services – APIs (4 of 6)

Miscellaneous APIs	Purpose
CFE_ES_BackgroundWakeup	Wakes up the CFE background 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_ProcessAsyncEvent	Notification that an asynchronous event was detected by the underlying OS/PSP

Critical Data Store APIs	Purpose
CFE_ES_RegisterCDS	Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
CFE_ES_GetCDSBlockIDByName	Get a CDS Block ID associated with a specified CDS Block name
CFE_ES_GetCDSBlockName	Get a Block name for a specified Block ID
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)

Executive Services – APIs (5 of 6)

Memory Manager APIs	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_PoolDelete	Deletes a memory pool that was previously created
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

Performance Monitor APIs	Purpose
CFE_ES_PerfLogEntry	Entry marker for use with Software Performance Analysis Tool.
CFE_ES_PerfLogExit	Exit marker for use with Software Performance Analysis Tool.
CFE_ES_PerfLogAdd	Adds a new entry to the data buffer

Executive Services – APIs (6 of 6)

Generic Counter APIs	Purpose
CFE_ES_RegisterGenCounter	This routine registers a generic thread-safe counter which can be used for inter-task management.
CFE_ES_DeleteGenCounter	This routine deletes a previously registered generic counter.
CFE_ES_IncrementGenCounter	This routine increments the specified generic counter.
CFE_ES_SetGenCount	This routine sets the specified generic counter to the specified value.
CFE_ES_GetGenCount	This routine gets the value of a generic counter.
CFE_ES_GetGenCounterIDByName	Get the Id associated with a generic counter name
CFE_ES_GetGenCounterName	Get a Counter name for a specified Counter ID

A Note on Resource IDs

- **cFS Caelum builds on the resource IDs present in previous versions**
- **Resource IDs are implemented as a separate module**
- **Resource IDs increase the type safety of cFE**
- **ES uses several Resource IDs extensively in its API calls:**
  - CFE_ES_AppId_t
  - CFE_ES_LibId_t
  - CFE_ES_TaskId_t
  - CFE_ES_CounterId_t
- **The ResourceID module provides utility functions to compare IDs and convert between integer types and ResourceIDs**

New in Caelum

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_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 – Platform 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_NONVOL_DISK_MOUNT_STRING	Default virtual path for persistent storage
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE	ES Volatile Startup Filename
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
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_POR_SYSLOG_MODE	Default System Log Mode following Power On Reset

Executive Services – Mission Configuration Parameters

Command List	Purpose
CFE_MISSION_ES_CDS_MAX_NAME_LENGTH	Maximum Length of CDS Name
CFE_MISSION_ES_DEFAULT_CRC	Mission Default CRC algorithm
CFE_MISSION_ES_MAX_APPLICATIONS	Mission Max Apps in a message
CFE_MISSION_ES_PERF_MAX_IDS	Define Max Number of Performance IDs for messages
CFE_MISSION_ES_POOL_MAX_BUCKETS	Maximum number of block sizes in pool structures
CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN	Maximum Length of Full CDS Name in messages

Executive Services – Platform Configuration Parameters

Command List	Purpose
CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE	Default System Log Mode following Processor Reset
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_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_MEMPOOL_ALIGN_SIZE_MIN	Define Memory Pool Alignment Size
CFE_PLATFORM_ES_POOL_MAX_BUCKETS	Maximum number of block sizes in pool structures
CFE_PLATFORM_ES_MAX_MEMORY_POOLS	Maximum number of memory pools
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

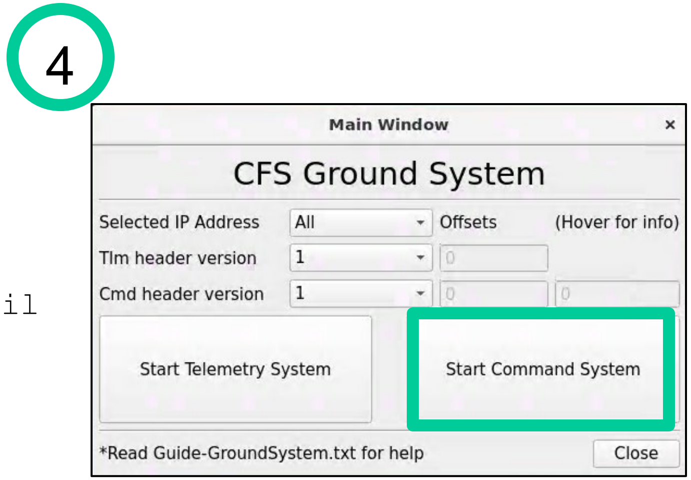
Part 1 – Start the Ground System

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 PyQt5 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 ../..
python3 GroundSystem.py
```

4. Select "Start Command System"



CFS Ground System

Selected IP Address: All — Offsets (Hover for info)

Tlm header version: 1 — 0

Cmd header version: 1 — 0 — 0

*Read Guide-GroundSystem.txt for help

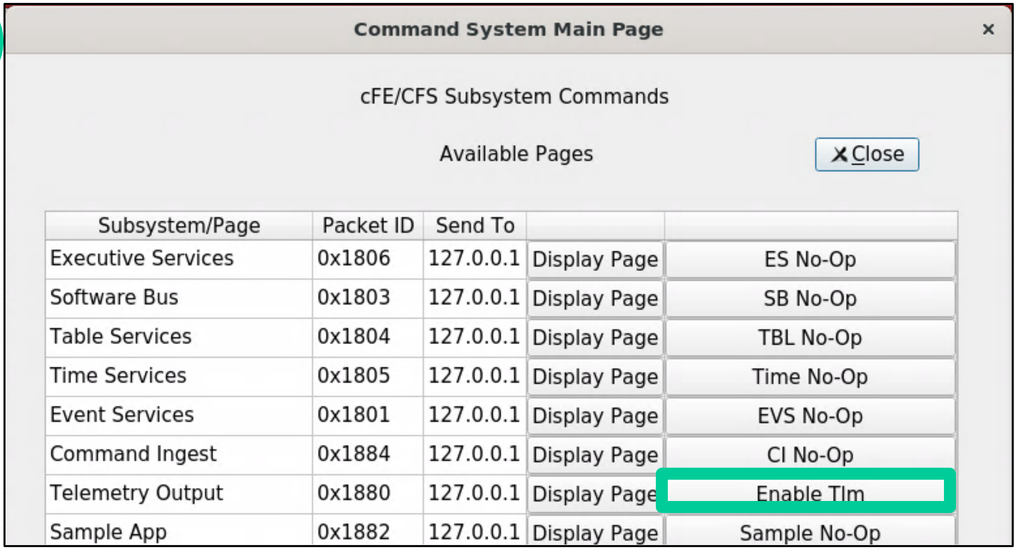
Exercise 2 - Command cFE Executive Service

Part 1 Continued

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

5



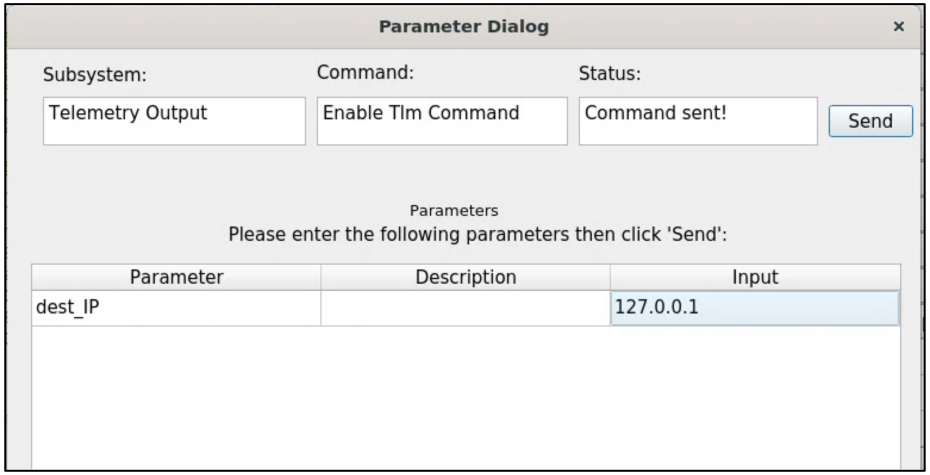
Command System Main Page ×

cFE/CFS Subsystem Commands

Available Pages ✗Close

Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op

6



Parameter Dialog ×

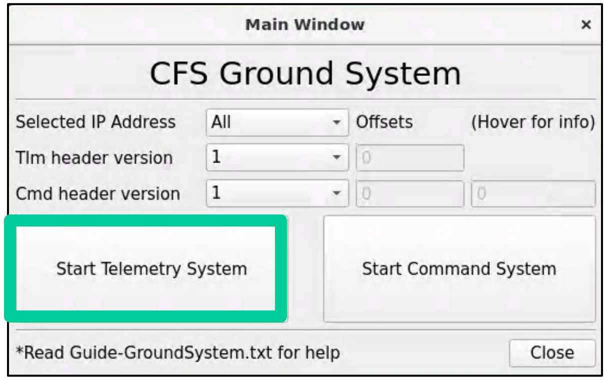
Subsystem:	Command:	Status:	
Telemetry Output	Enable Tlm Command	Command sent!	Send

Parameters

Please enter the following parameters then click 'Send':

Parameter	Description	Input
dest_IP		127.0.0.1

7



Main Window ×

CFS Ground System

Selected IP Address	All	Offsets	(Hover for info)
Tlm header version	1	0	
Cmd header version	1	0	0

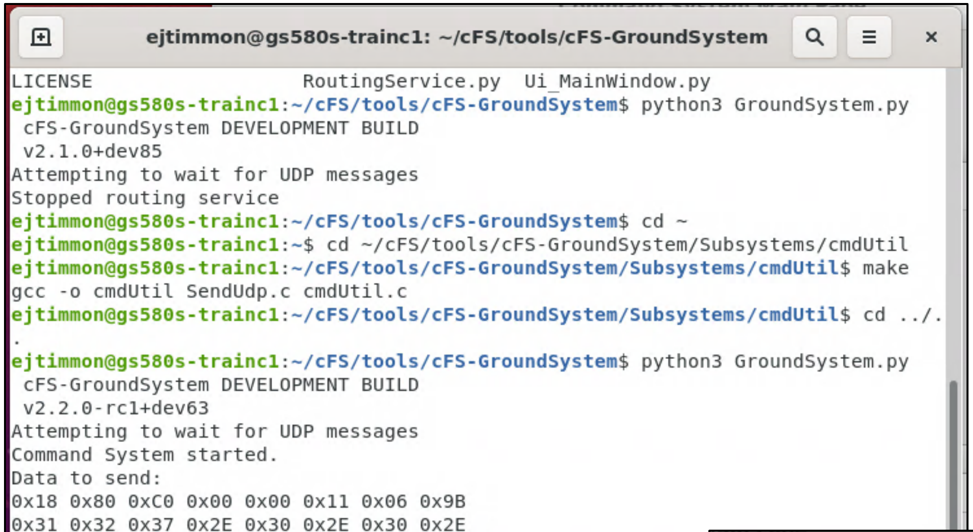
Start Telemetry System

Start Command System

*Read Guide-GroundSystem.txt for help

Close

Exercise 2 – Part 1 Recap

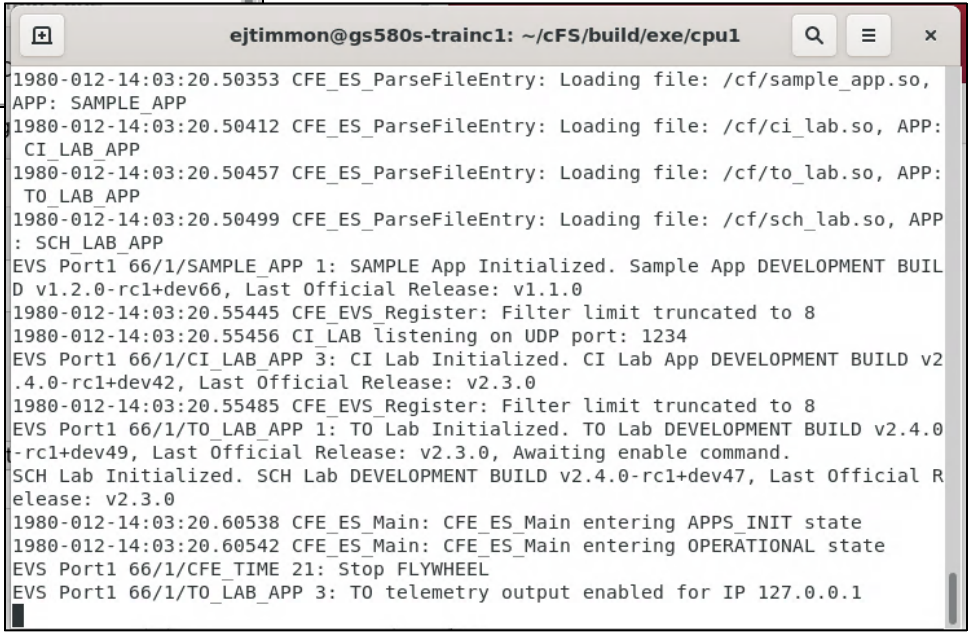


```
LICENSE                    RoutingService.py  Ui_MainWindow.py
ejtimmon@gs580s-trainc1:~/cFS/tools/cFS-GroundSystem$ python3 GroundSystem.py
 cFS-GroundSystem DEVELOPMENT BUILD
 v2.1.0+dev85
Attempting to wait for UDP messages
Stopped routing service
ejtimmon@gs580s-trainc1:~/cFS/tools/cFS-GroundSystem$ cd ~
ejtimmon@gs580s-trainc1:~$ cd ~/cFS/tools/cFS-GroundSystem/Subsystems/cmdUtil
ejtimmon@gs580s-trainc1:~/cFS/tools/cFS-GroundSystem/Subsystems/cmdUtil$ make
gcc -o cmdUtil SendUdp.c cmdUtil.c
ejtimmon@gs580s-trainc1:~/cFS/tools/cFS-GroundSystem/Subsystems/cmdUtil$ cd ../.
.
ejtimmon@gs580s-trainc1:~/cFS/tools/cFS-GroundSystem$ python3 GroundSystem.py
 cFS-GroundSystem DEVELOPMENT BUILD
 v2.2.0-rc1+dev63
Attempting to wait for UDP messages
Command System started.
Data to send:
0x18 0x80 0xC0 0x00 0x00 0x11 0x06 0x9B
0x31 0x32 0x37 0x2E 0x30 0x2E 0x30 0x2E
0x31 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Detected Spacecraft1 at 127.0.0.1
```

Python GUI Terminal Window

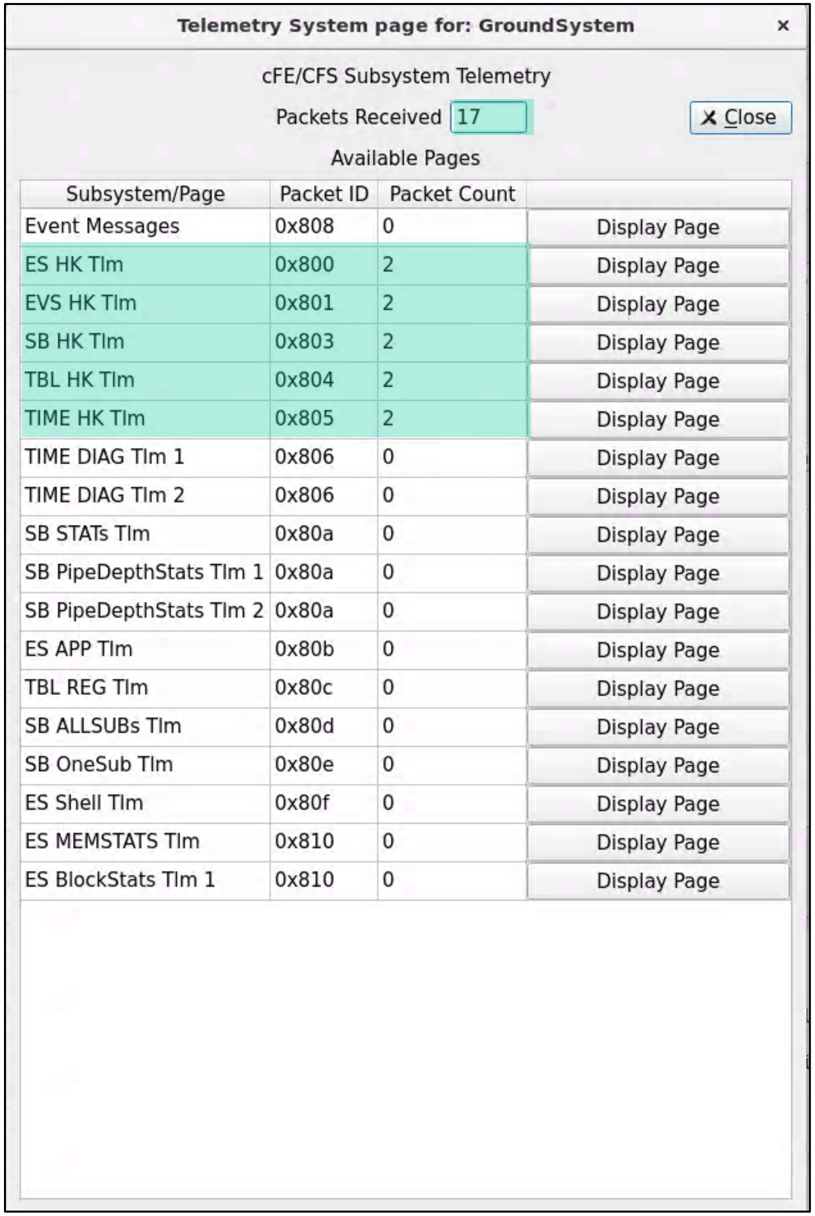
cFS Terminal Window



```
1980-012-14:03:20.50353 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so,
APP: SAMPLE_APP
1980-012-14:03:20.50412 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP:
 CI_LAB_APP
1980-012-14:03:20.50457 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP:
 TO_LAB_APP
1980-012-14:03:20.50499 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP
: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUIL
D v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-14:03:20.55445 CFE_EVS_Register: Filter limit truncated to 8
1980-012-14:03:20.55456 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2
.4.0-rc1+dev42, Last Official Release: v2.3.0
1980-012-14:03:20.55485 CFE_EVS_Register: Filter limit truncated to 8
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0
-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable command.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official R
elease: v2.3.0
1980-012-14:03:20.60538 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.60542 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 66/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
```

Exercise 2 – Part 1 Recap

After Step 7, cFE housekeeping packet counts should start incrementing



Telemetry System page for: GroundSystem

cFE/CFS Subsystem Telemetry

Packets Received 17

Available Pages

Subsystem/Page	Packet ID	Packet Count	
Event Messages	0x808	0	Display Page
ES HK Tlm	0x800	2	Display Page
EVS HK Tlm	0x801	2	Display Page
SB HK Tlm	0x803	2	Display Page
TBL HK Tlm	0x804	2	Display Page
TIME HK Tlm	0x805	2	Display Page
TIME DIAG Tlm 1	0x806	0	Display Page
TIME DIAG Tlm 2	0x806	0	Display Page
SB STATs Tlm	0x80a	0	Display Page
SB PipeDepthStats Tlm 1	0x80a	0	Display Page
SB PipeDepthStats Tlm 2	0x80a	0	Display Page
ES APP Tlm	0x80b	0	Display Page
TBL REG Tlm	0x80c	0	Display Page
SB ALLSUBs Tlm	0x80d	0	Display Page
SB OneSub Tlm	0x80e	0	Display Page
ES Shell Tlm	0x80f	0	Display Page
ES MEMSTATS Tlm	0x810	0	Display Page
ES BlockStats Tlm 1	0x810	0	Display Page

Exercise 2 - Command cFE Executive Service

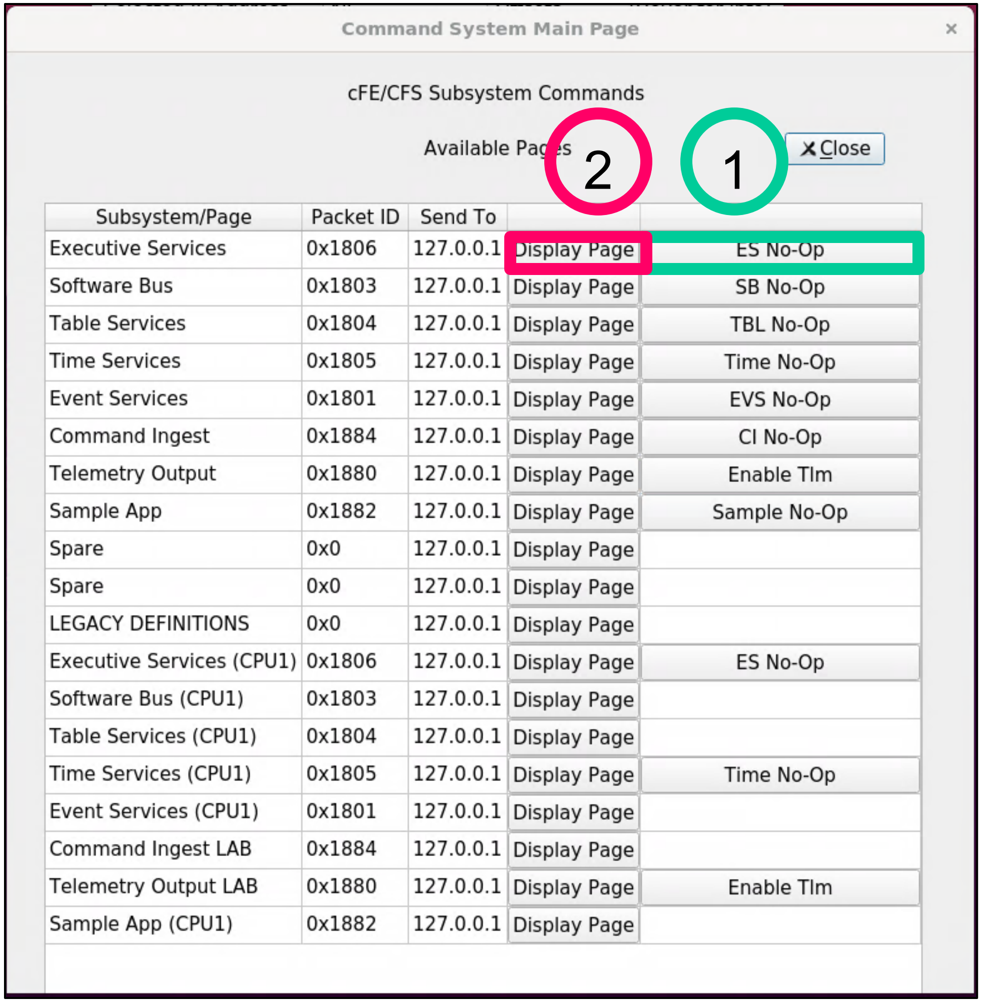
Part 2 – Command Executive Services

Send a No-Op Command

1. On the Command System Main Page, select "ES No-Op".
   - A no-op message should appear in the cFS screen.

Restart an application

2. On the Command System Main Page, click the "Display Page" button beside "Executive Services CPU1".



cFE/CFS Subsystem Commands

Available Pages

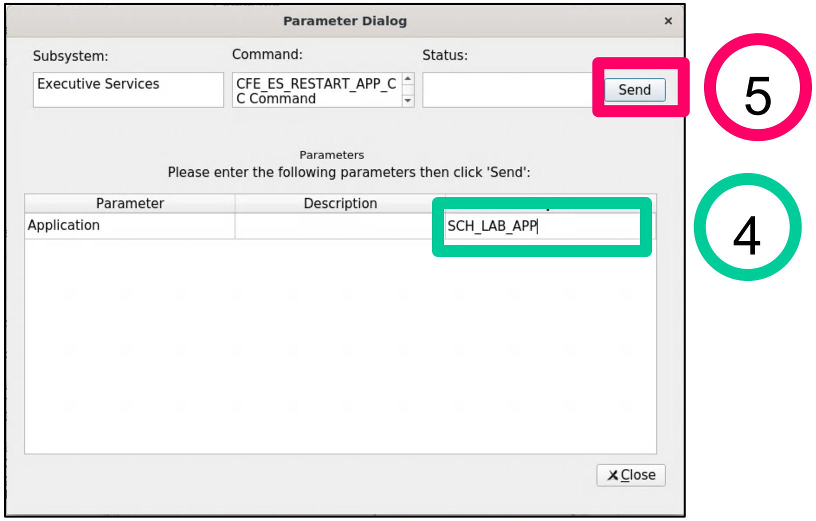
Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	
Spare	0x0	127.0.0.1	Display Page	
LEGACY DEFINITIONS	0x0	127.0.0.1	Display Page	
Executive Services (CPU1)	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus (CPU1)	0x1803	127.0.0.1	Display Page	
Table Services (CPU1)	0x1804	127.0.0.1	Display Page	
Time Services (CPU1)	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services (CPU1)	0x1801	127.0.0.1	Display Page	
Command Ingest LAB	0x1884	127.0.0.1	Display Page	
Telemetry Output LAB	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App (CPU1)	0x1882	127.0.0.1	Display Page	

Exercise 2 - Command cFE Executive Service

Part 2 – Command Executive Services - Continued

3. Click the "Send" button beside "CFE_ES_RESTART_APP_CC".
4. Enter "SCH_LAB_APP" in the "Input" field.
5. 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



Parameter Dialog

Subsystem: Executive Services

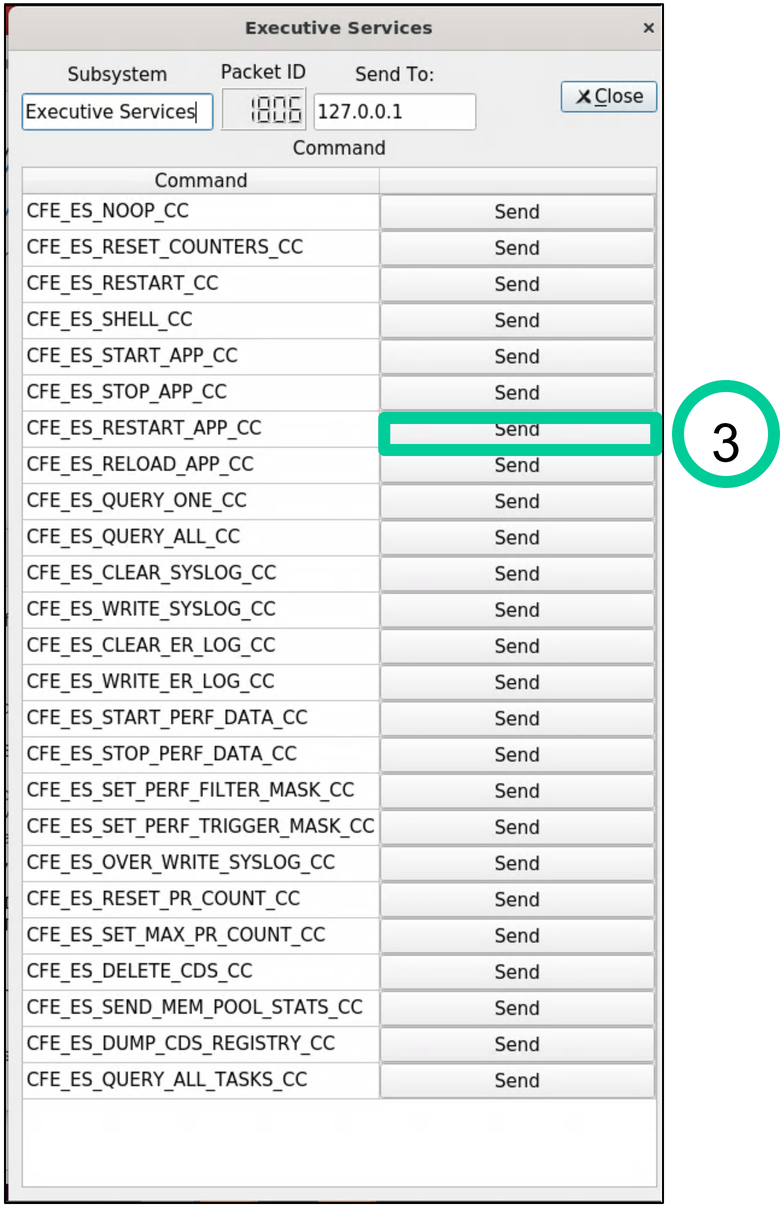
Command: CFE_ES_RESTART_APP_CC Command

Status:

Parameters

Please enter the following parameters then click 'Send':

Parameter	Description	
Application		SCH_LAB_APP



Executive Services

Subsystem: Executive Services

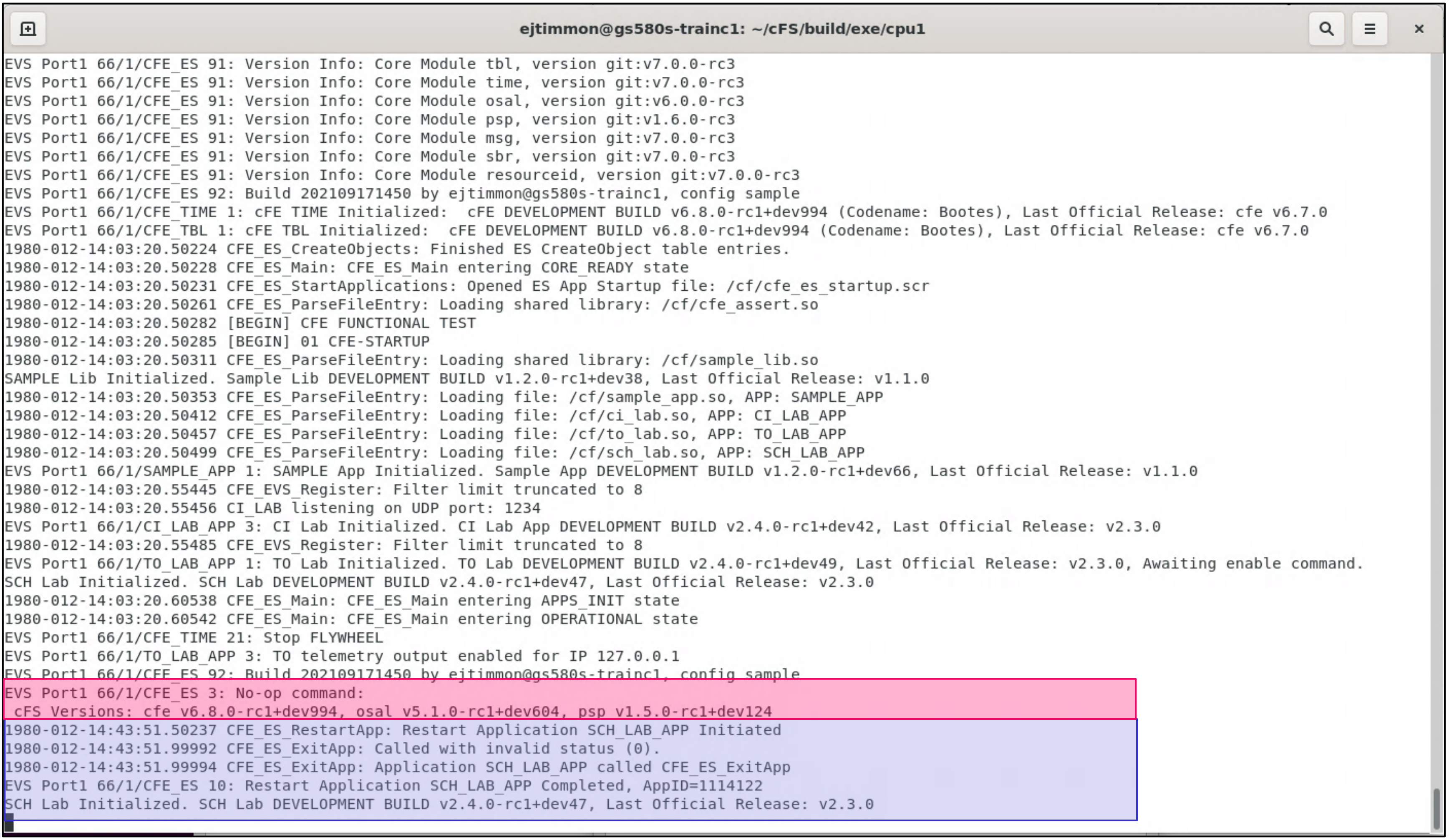
Packet ID: 1806

Send To: 127.0.0.1

Command

Command	
CFE_ES_NOOP_CC	Send
CFE_ES_RESET_COUNTERS_CC	Send
CFE_ES_RESTART_CC	Send
CFE_ES_SHELL_CC	Send
CFE_ES_START_APP_CC	Send
CFE_ES_STOP_APP_CC	Send
CFE_ES_RESTART_APP_CC	Send
CFE_ES_RELOAD_APP_CC	Send
CFE_ES_QUERY_ONE_CC	Send
CFE_ES_QUERY_ALL_CC	Send
CFE_ES_CLEAR_SYSLOG_CC	Send
CFE_ES_WRITE_SYSLOG_CC	Send
CFE_ES_CLEAR_ER_LOG_CC	Send
CFE_ES_WRITE_ER_LOG_CC	Send
CFE_ES_START_PERF_DATA_CC	Send
CFE_ES_STOP_PERF_DATA_CC	Send
CFE_ES_SET_PERF_FILTER_MASK_CC	Send
CFE_ES_SET_PERF_TRIGGER_MASK_CC	Send
CFE_ES_OVER_WRITE_SYSLOG_CC	Send
CFE_ES_RESET_PR_COUNT_CC	Send
CFE_ES_SET_MAX_PR_COUNT_CC	Send
CFE_ES_DELETE_CDS_CC	Send
CFE_ES_SEND_MEM_POOL_STATS_CC	Send
CFE_ES_DUMP_CDS_REGISTRY_CC	Send
CFE_ES_QUERY_ALL_TASKS_CC	Send

Exercise 2 Part 2 Recap



ejtimmon@gs580s-trainc1: ~/cFS/build/exe/cpu1

```
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module tbl, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module time, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module osal, version git:v6.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module psp, version git:v1.6.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module msg, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sbr, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module resourceid, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_TIME 1: cFE TIME Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 1: cFE TBL Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
1980-012-14:03:20.50224 CFE_ES_CreateObjects: Finished ES CreateObject table entries.
1980-012-14:03:20.50228 CFE_ES_Main: CFE_ES_Main entering CORE_READY state
1980-012-14:03:20.50231 CFE_ES_StartApplications: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.50261 CFE_ES_ParseFileEntry: Loading shared library: /cf/cfe_assert.so
1980-012-14:03:20.50282 [BEGIN] CFE FUNCTIONAL TEST
1980-012-14:03:20.50285 [BEGIN] 01 CFE-STARTUP
1980-012-14:03:20.50311 CFE_ES_ParseFileEntry: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.2.0-rc1+dev38, Last Official Release: v1.1.0
1980-012-14:03:20.50353 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.50412 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.50457 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.50499 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-14:03:20.55445 CFE_EVS_Register: Filter limit truncated to 8
1980-012-14:03:20.55456 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.4.0-rc1+dev42, Last Official Release: v2.3.0
1980-012-14:03:20.55485 CFE_EVS_Register: Filter limit truncated to 8
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable command.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
1980-012-14:03:20.60538 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.60542 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 66/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_ES 3: No-op command:
 cFS Versions: cfe v6.8.0-rc1+dev994, osal v5.1.0-rc1+dev604, psp v1.5.0-rc1+dev124
1980-012-14:43:51.50237 CFE_ES_RestartApp: Restart Application SCH_LAB_APP Initiated
1980-012-14:43:51.99992 CFE_ES_ExitApp: Called with invalid status (0).
1980-012-14:43:51.99994 CFE_ES_ExitApp: Application SCH_LAB_APP called CFE_ES_ExitApp
EVS Port1 66/1/CFE_ES 10: Restart Application SCH_LAB_APP Completed, AppID=1114122
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
```

ES No-Op Command

ES Restart App Command

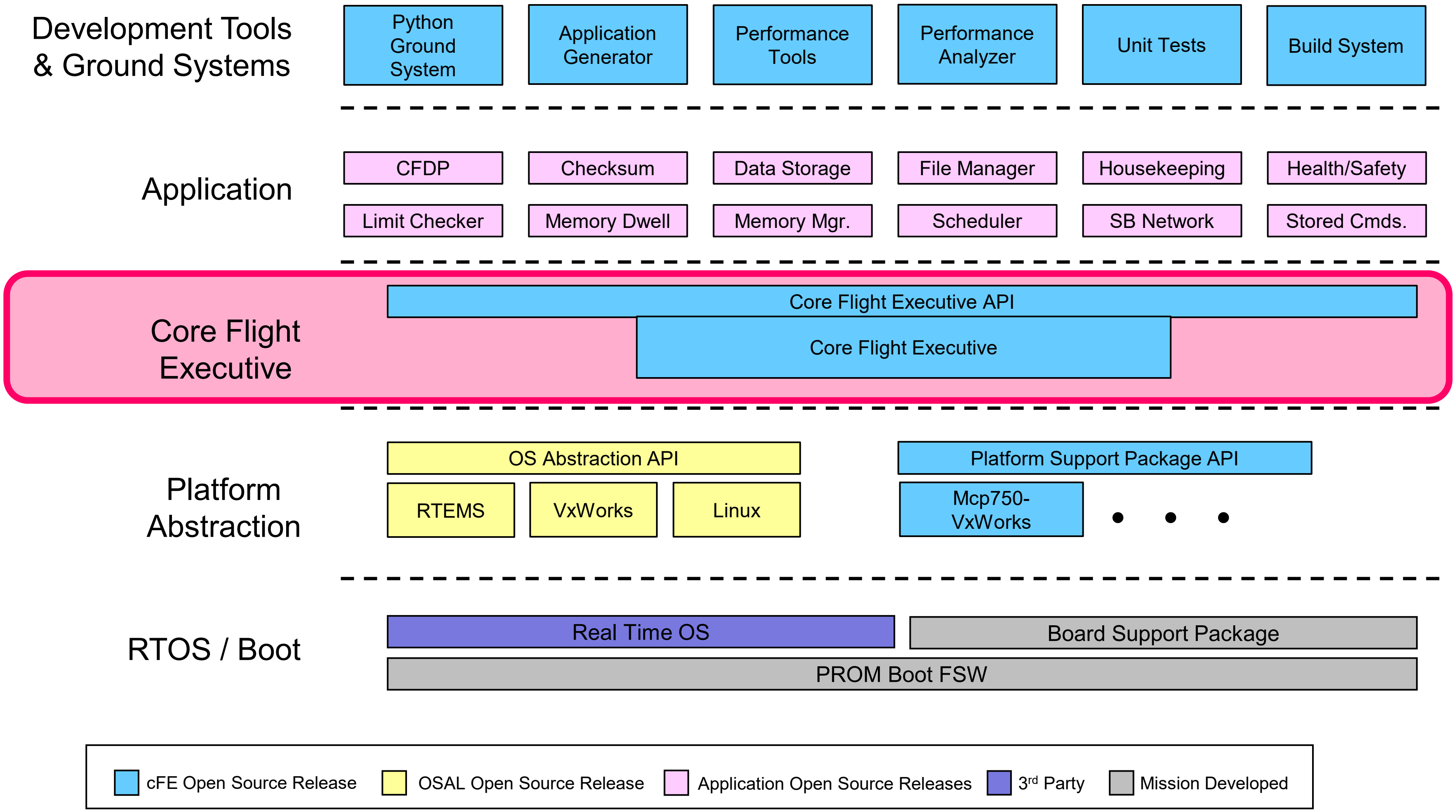
Core Flight System (cFS) Training

Module 2b: Software Bus Services

Course Agenda

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

Software Bus - cFS Context



Development Tools & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

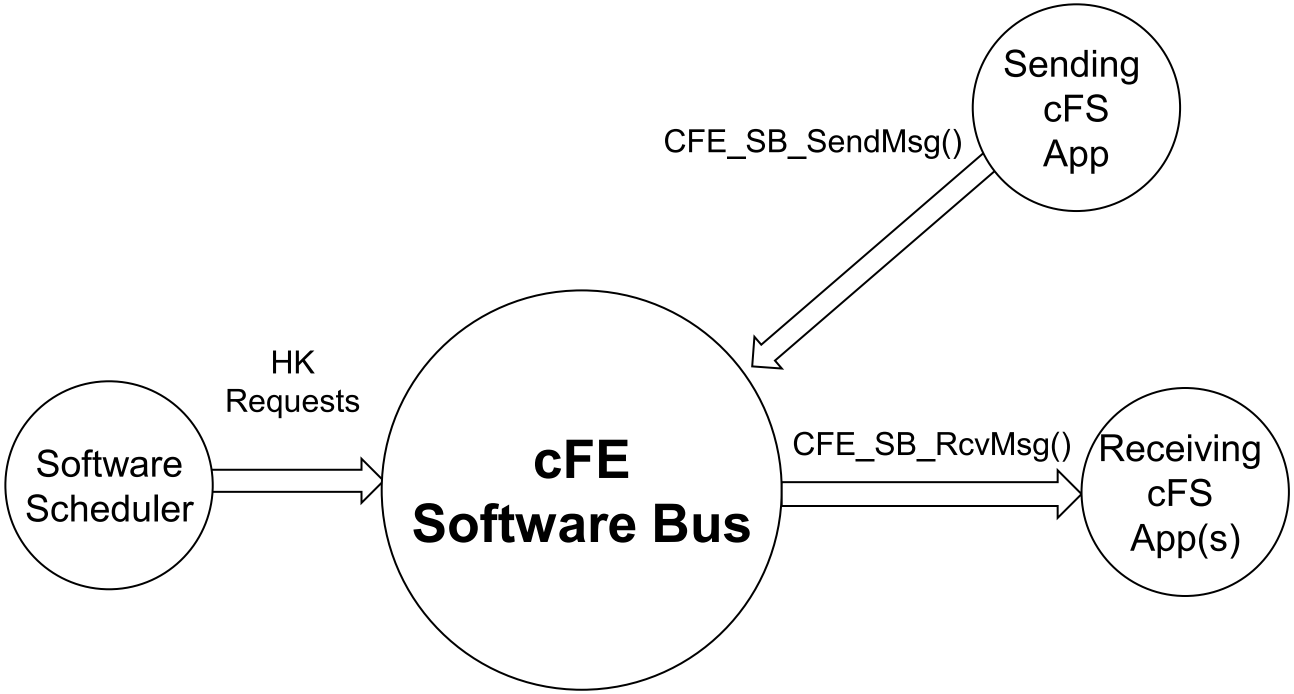
PROM Boot FSW

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

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 Terms

- **Pipe – Destination to which SB Messages are sent; queues that can hold SB Messages until they are read out and processed**
- **Message – A collection of data treated as a single entity.**
- **Buffer – The generic piece of data moved on the Software Bus**
  - Alignment is enforced at the buffer level
  - In general, applications receive buffers and cast them to a specific message type to use them

Software Bus and Message Module

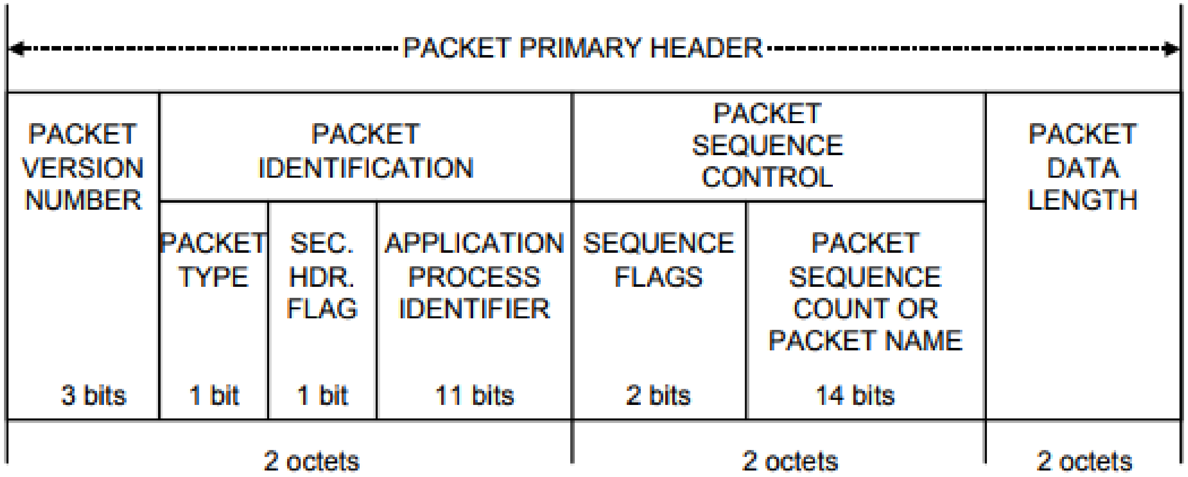
- cFS Caelum introduces a Message Module that encapsulates the definition of messages passed by cFE SB
- cFE SB handles the routing of messages
- cFE Message Module handles the definition and parsing of individual messages

New in Caelum

Software Bus – Messages (1 of 2)

- **Messages are routed by a “MessageID”**
  - This should always be treated as opaque – applications should not try to directly access the fields of a MessageID
  - By default, the Message Module provides two implementations (MISSION_MSG_V1 and MISSION_MSG_V2)
    - MISSION_MSG_V1 maps directly to the CCSDS Stream ID

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



<table>
<tr><th colspan="7">PACKET PRIMARY HEADER</th></tr>
<tr><td rowspan="2">PACKET VERSION NUMBER</td><td colspan="3">PACKET IDENTIFICATION</td><td colspan="2">PACKET SEQUENCE CONTROL</td><td rowspan="2">PACKET DATA LENGTH</td></tr>
<tr><td>PACKET TYPE</td><td>SEC. HDR. FLAG</td><td>APPLICATION PROCESS IDENTIFIER</td><td>SEQUENCE FLAGS</td><td>PACKET SEQUENCE COUNT OR PACKET NAME</td></tr>
<tr><td>3 bits</td><td>1 bit</td><td>1 bit</td><td>11 bits</td><td>2 bits</td><td>14 bits</td><td></td></tr>
<tr><td colspan="4">2 octets</td><td colspan="2">2 octets</td><td>2 octets</td></tr>
</table>

Software Bus – Messages (2 of 2)

- **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

- **Message formats are defined in the Message Module, along with functions to access message header fields (CFE_MSG_GetApId, CFE_MSG_GetSequenceCount, etc.)**

```
union CFE_MSG_Message {
    CCSDS_SpacePacket_t CCSDS;
    uint8               Byte[sizeof(CCSDS_SpacePacket_t)];
}
struct CFE_MSG_CommandHeader {
    CFE_MSG_Message_t                Msg;
    CFE_MSG_CommandSecondaryHeader_t Sec;
}
struct CFE_MSG_TelemetryHeader {
    CFE_MSG_Message_t                  Msg;
    CFE_MSG_TelemetrySecondaryHeader_t Sec;
    uint8                              Spare[4];
}
```

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

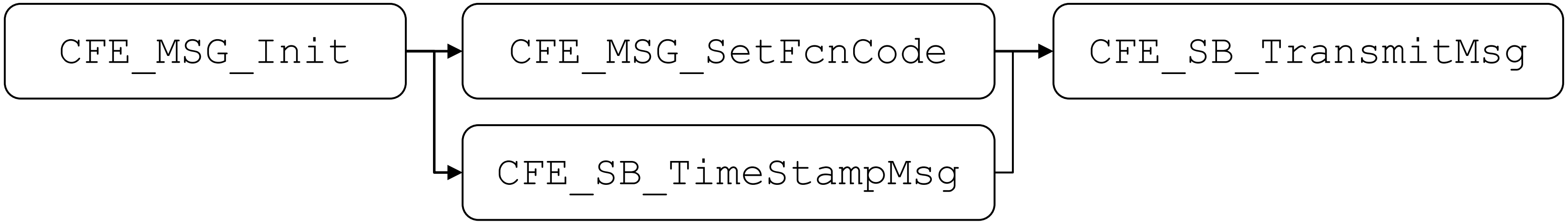
- **Telemetry**
  - Housekeeping Status
    - Counters (No subscribers, send errors, pipe overflows, etc.), Memory Stats
- **Telemetry packets generated by command**
  - Statistics
  - Subscription Report
- **Files generated by command**
  - Routing Info
  - Pipe Info
  - Message ID to Route

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 2)

- **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:**



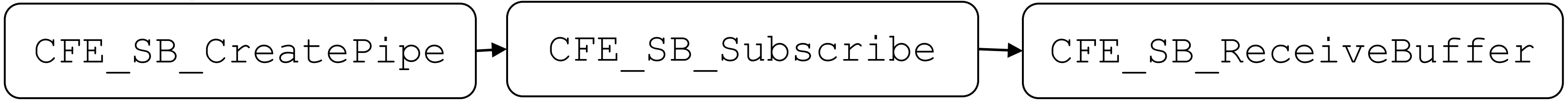
CFE_MSG_Init

CFE_MSG_SetFcnCode

CFE_SB_TimeStampMsg

CFE_SB_TransmitMsg

- **Receiving Messages:**



CFE_SB_CreatePipe

CFE_SB_Subscribe

CFE_SB_ReceiveBuffer

Software Bus – App Development (2 of 2)

- **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**

```
CFE_Status_t CFE_SB_ReceiveBuffer(CFE_SB_Buffer_t **BufPtr,
                                  CFE_SB_PipeId_t PipeId,
                                  int32 TimeOut);
```

cFE Software Bus APIs

Pipe Management APIs	Purpose
CFE_SB_CreatePipe	Creates a new software bus pipe.
CFE_SB_DeletePipe	Delete a software bus pipe.
CFE_SB_PipeId_ToIndex	Obtain an index value correlating to an SB Pipe ID
CFE_SB_SetPipeOpts	Set options on a pipe.
CFE_SB_GetPipeOpts	Get options on a pipe.
CFE_SB_GetPipeName	Get the pipe name for a given id.
CFE_SB_GetPipeIdByName	Get pipe id by pipe name.

Message Subscription Control APIs	Purpose
CFE_SB_SubscribeEx	Subscribe to a message on the software bus
CFE_SB_Subscribe	Subscribe to a message on the software bus with default parameters
CFE_SB_SubscribeLocal	Subscribe to a message while keeping the request local to a CPU
CFE_SB_Unsubscribe	Remove a subscription to a message on the software bus
CFE_SB_UnsubscribeLocal	Remove a subscription to a message on the software bus on the current CPU

cFE Software Bus APIs

Send/Receive Message APIs	Purpose
CFE_SB_TransmitMsg	Transmit a message
CFE_SB_ReceiveBuffer	Receive a message from a software bus pipe

Zero Copy APIs	Purpose
CFE_SB_AllocateMessageBuffer	Get a buffer pointer to use for "zero copy" SB sends.
CFE_SB_ReleaseMessageBuffer	Release an unused "zero copy" buffer pointer.
CFE_SB_TransmitBuffer	Transmit a buffer

Message Characteristics APIs	Purpose
CFE_SB_SetUserDataLength	Sets the length of user data in a software bus message.
CFE_SB_TimeStampMsg	Sets the time field in a software bus message with the current spacecraft time.
CFE_SB_MessageStringSet	Copies a string into a software bus message
CFE_SB_GetUserData	Get a pointer to the user data portion of a software bus message.
CFE_SB_GetUserDataLength	Gets the length of user data in a software bus message.
CFE_SB_MessageStringGet	Copies a string out of a software bus message

cFE Software Bus APIs

Message ID APIs	Purpose
CFE_SB_IsValidMsgId	Identifies whether a given CFE_SB_MsgId_t is valid
CFE_SB_MsgId_Equal	Identifies whether two #CFE_SB_MsgId_t values are equal
CFE_SB_MsgIdToValue	Converts a #CFE_SB_MsgId_t to a normal integer
CFE_SB_ValueToMsgId	Converts a normal integer into a #CFE_SB_MsgId_t

cFE Message Module APIs

Generic Message APIs	Purpose
CFE_MSG_Init	Initialize a message

Message Primary Header APIs	Purpose
CFE_MSG_GetSize	Gets the total size of a message.
CFE_MSG_SetSize	Sets the total size of a message.
CFE_MSG_GetType	Gets the message type.
CFE_MSG_SetType	Sets the message type.
CFE_MSG_GetHeaderVersion	Gets the message header version.
CFE_MSG_SetHeaderVersion	Sets the message header version.
CFE_MSG_GetHasSecondaryHeader	Gets the message secondary header boolean
CFE_MSG_SetHasSecondaryHeader	Sets the message secondary header boolean
CFE_MSG_GetApId	Gets the message application ID
CFE_MSG_SetApId	Sets the message application ID
CFE_MSG_GetSegmentationFlag	Gets the message segmentation flag
CFE_MSG_SetSegmentationFlag	Sets the message segmentation flag
CFE_MSG_GetSequenceCount	Gets the message sequence count
CFE_MSG_SetSequenceCount	Sets the message sequence count
CFE_MSG_GetNextSequenceCount	Gets the next sequence count value (rolls over if appropriate)

cFE Message Module APIs

Message Extended Header APIs	Purpose
CFE_MSG_GetEDSVersion	Gets the message EDS version
CFE_MSG_SetEDSVersion	Sets the message EDS version
CFE_MSG_GetEndian	Gets the message endian
CFE_MSG_SetEndian	Sets the message endian
CFE_MSG_GetPlaybackFlag	Gets the message playback flag
CFE_MSG_SetPlaybackFlag	Sets the message playback flag
CFE_MSG_GetSubsystem	Gets the message subsystem
CFE_MSG_SetSubsystem	Sets the message subsystem
CFE_MSG_GetSystem	Gets the message system
CFE_MSG_SetSystem	Sets the message system

Message Secondary Header APIs	Purpose
CFE_MSG_GenerateChecksum	Calculates and sets the checksum of a message
CFE_MSG_ValidateChecksum	Validates the checksum of a message.
CFE_MSG_SetFcnCode	Sets the function code field in a message.
CFE_MSG_GetFcnCode	Gets the function code field from a message.
CFE_MSG_GetMsgTime	Gets the time field from a message.
CFE_MSG_SetMsgTime	Sets the time field in a message.

cFE Message Module APIs

Message Id APIs	Purpose
CFE_MSG_GetMsgId	Gets the message id from a message.
CFE_MSG_SetMsgId	Sets the message id bits in a message.
CFE_MSG_GetTypeFromMsgId	Gets message type using message ID

cFE Software Bus Command List

SB Command List	Purpose
CFE_SB_NoopCmd	Software Bus No-Op
CFE_SB_ResetCountersCmd	Resets counters in the Software Bus housekeeping telemetry
CFE_SB_EnableSubReportingCmd	Enable Subscription Reporting Command
CFE_SB_DisableSubReportingCmd	Disable Subscription Reporting Command
CFE_SB_SendHKTlmCmd	Function to send the SB housekeeping packet
CFE_SB_EnableRouteCmd	Enable Software Bus Route
CFE_SB_DisableRouteCmd	Disable Software Bus Route
CFE_SB_SendStatsCmd	Send Software Bus Statistics
CFE_SB_WriteRoutingInfoCmd	Write Software Bus Routing Info to a File
CFE_SB_WritePipeInfoCmd	Write Pipe Info to a File
CFE_SB_WriteMapInfoCmd	Write Map Info to a File
CFE_SB_SendPrevSubsCmd	Generates a series of packets that contain information regarding all subscriptions previously received by SB.

Software Bus – Platform 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_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_START_TASK_PRIORITY	SB Task Priority
CFE_PLATFORM_SB_START_TASK_STACK_SIZE	SB Task Stack Size

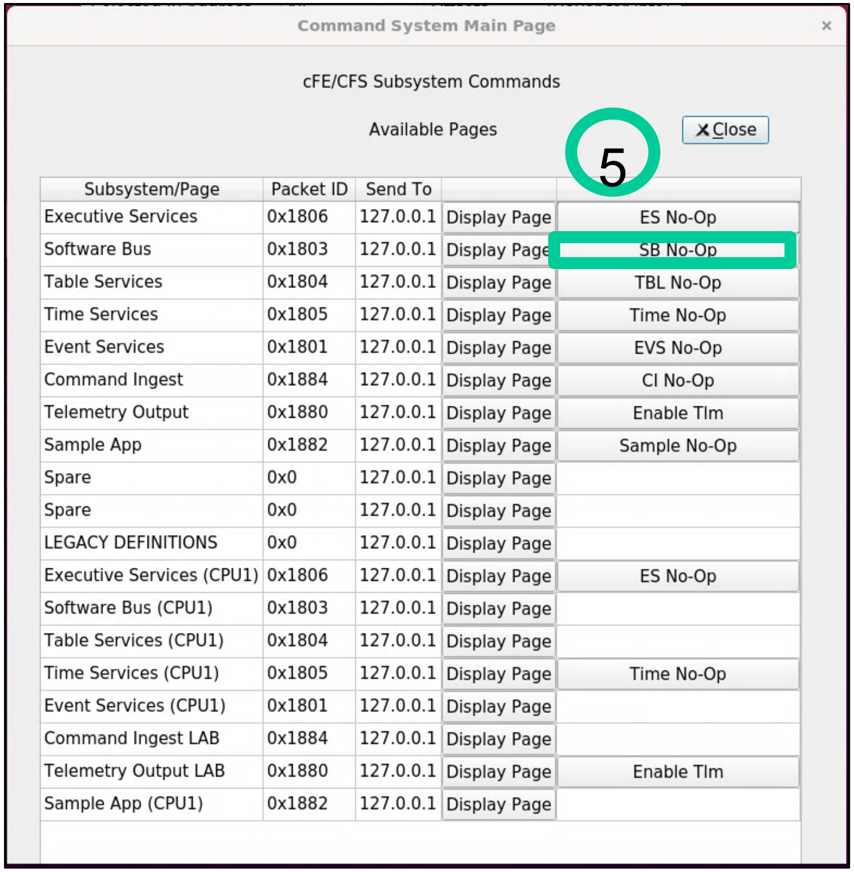
Software Bus – Mission Configuration Parameters

Parameter	Purpose
CFE_MISSION_SB_MAX_SB_MSG_SIZE	Maximum SB Message Size
CFE_MISSION_SB_MAX_PIPES	Maximum Number of pipes that SB command/telemetry messages may hold

Exercise 3 - Command cFE Software Bus

Part 1 – Send a No-Op Command

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 “SB No-Op" button beside "Software Bus"
    - Click the "Send" button beside "Software Bus No-Op"
    - Click "Send"



Command System Main Page ×

cFE/CFS Subsystem Commands

Available Pages

✕Close

5

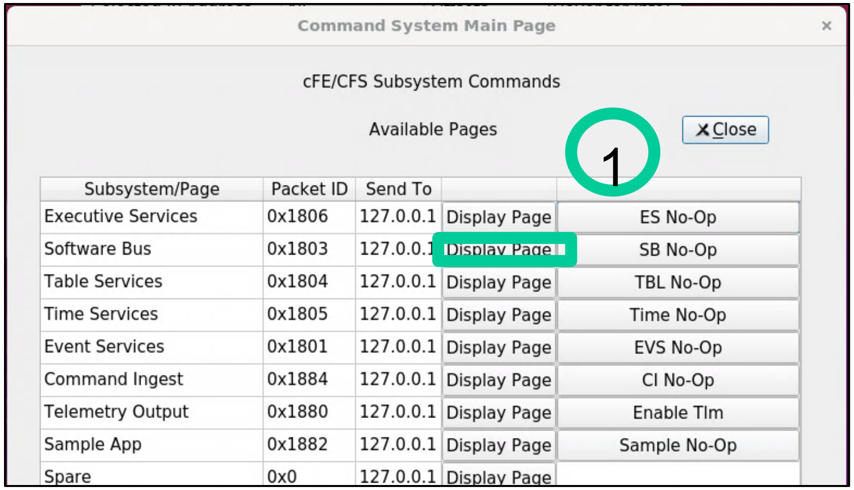
Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	
Spare	0x0	127.0.0.1	Display Page	
LEGACY DEFINITIONS	0x0	127.0.0.1	Display Page	
Executive Services (CPU1)	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus (CPU1)	0x1803	127.0.0.1	Display Page	
Table Services (CPU1)	0x1804	127.0.0.1	Display Page	
Time Services (CPU1)	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services (CPU1)	0x1801	127.0.0.1	Display Page	
Command Ingest LAB	0x1884	127.0.0.1	Display Page	
Telemetry Output LAB	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App (CPU1)	0x1882	127.0.0.1	Display Page	

Exercise 3 - Command cFE Software Bus

Part 2 – Write the Routing Map

1. Click the “Display Page” button beside “Software Bus”
2. In the "Software Bus" window, click the "Send" button beside “CFE_SEND_MAP_INFO_CC"
3. Enter "/cf/map.bin" in the "Input" field next to "Filename"
4. 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

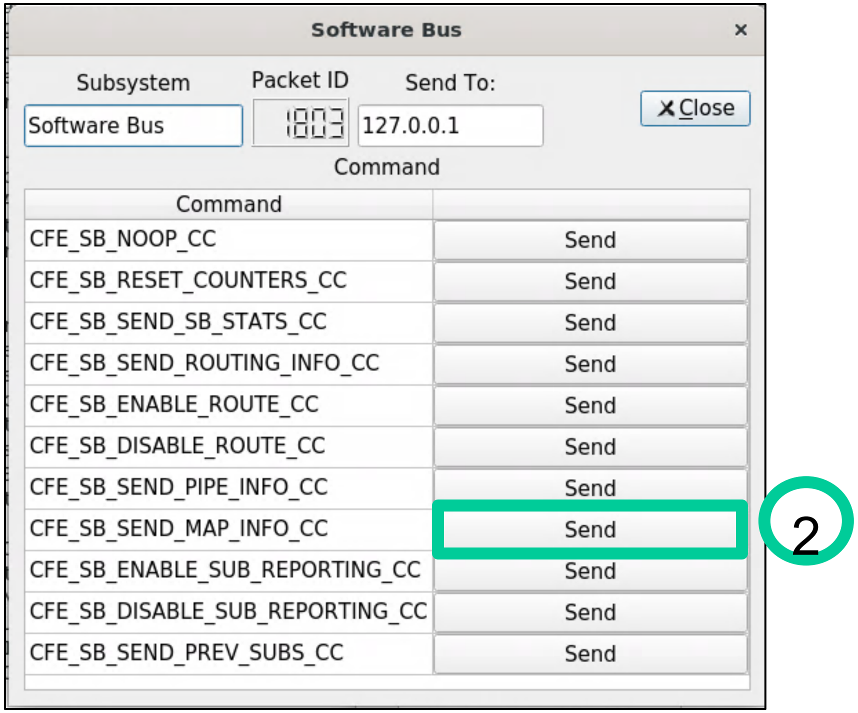


Command System Main Page

cFE/CFS Subsystem Commands

Available Pages

Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	



Software Bus

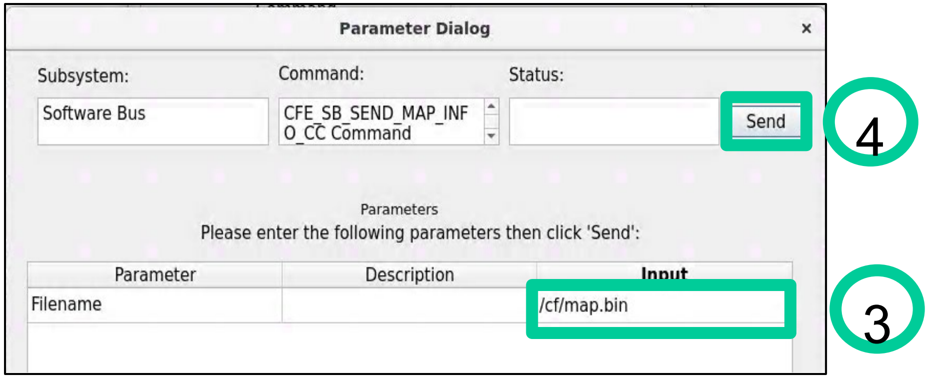
Subsystem: Software Bus

Packet ID: 1803

Send To: 127.0.0.1

Command

Command	
CFE_SB_NOOP_CC	Send
CFE_SB_RESET_COUNTERS_CC	Send
CFE_SB_SEND_SB_STATS_CC	Send
CFE_SB_SEND_ROUTING_INFO_CC	Send
CFE_SB_ENABLE_ROUTE_CC	Send
CFE_SB_DISABLE_ROUTE_CC	Send
CFE_SB_SEND_PIPE_INFO_CC	Send
CFE_SB_SEND_MAP_INFO_CC	Send
CFE_SB_ENABLE_SUB_REPORTING_CC	Send
CFE_SB_DISABLE_SUB_REPORTING_CC	Send
CFE_SB_SEND_PREV_SUBS_CC	Send



Parameter Dialog

Subsystem: Software Bus

Command: CFE_SB_SEND_MAP_INFO_CC Command

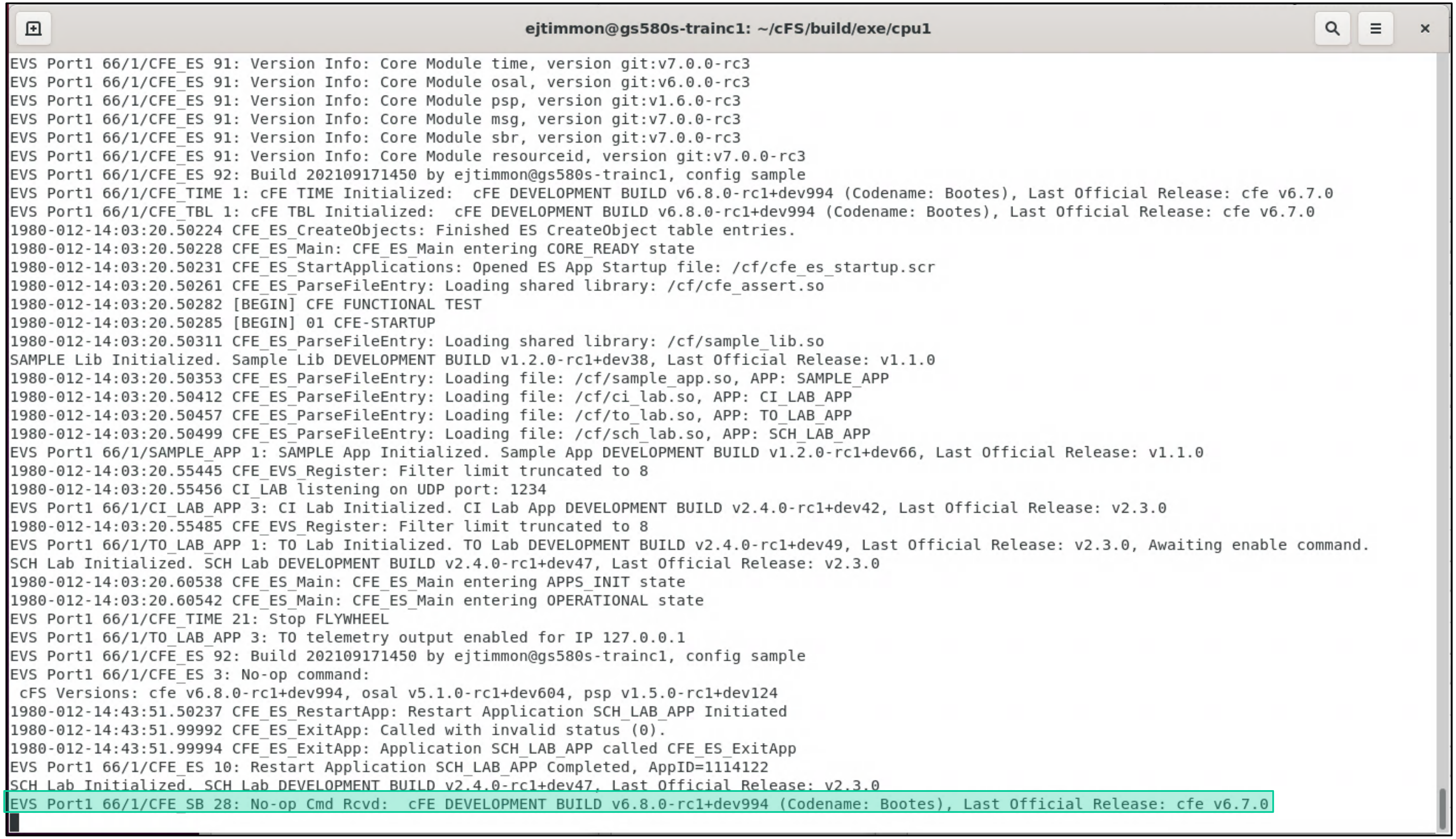
Status:

Parameters

Please enter the following parameters then click 'Send':

Parameter	Description	Input
Filename		/cf/map.bin

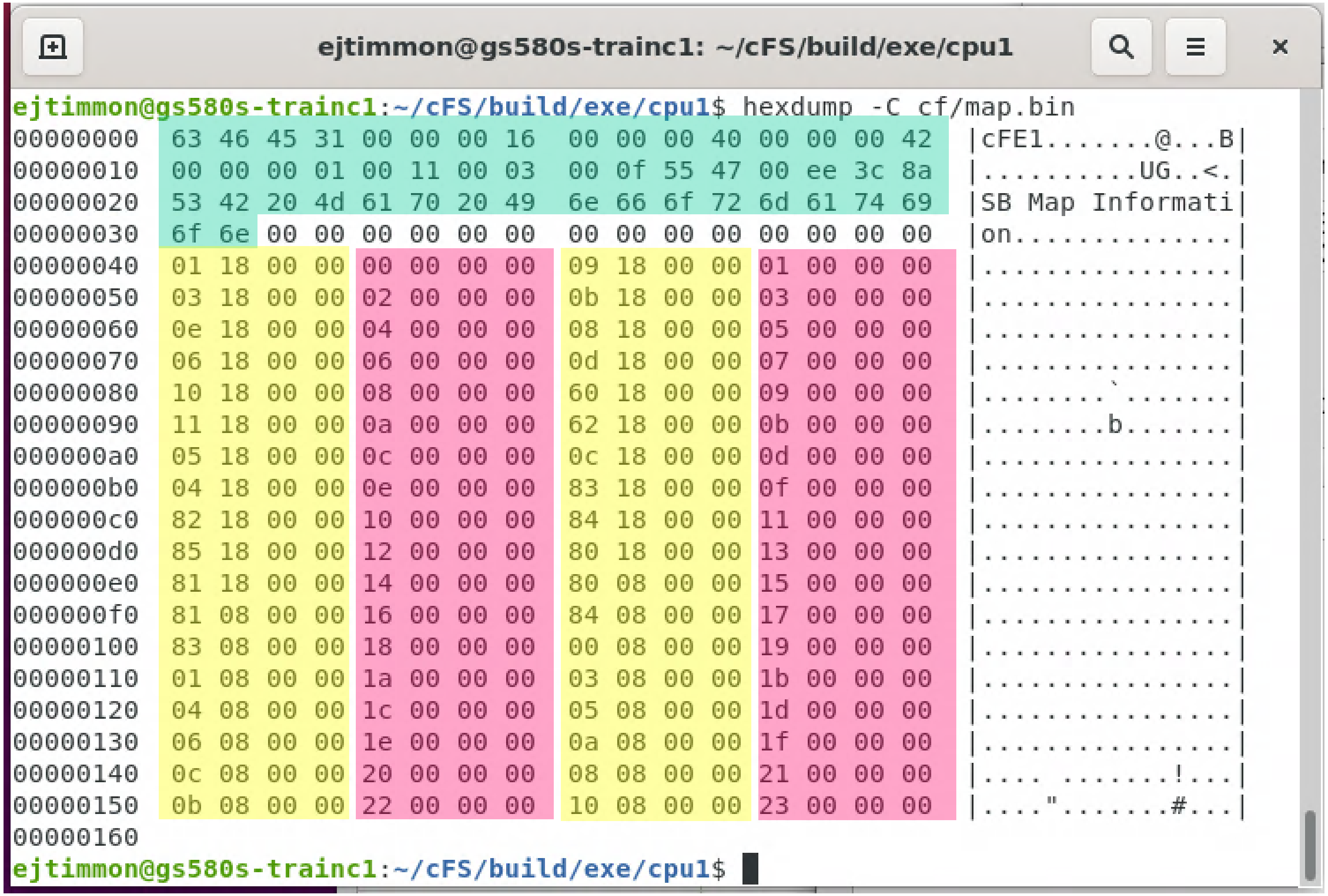
Exercise 3 Recap



```
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module time, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module osal, version git:v6.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module psp, version git:v1.6.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module msg, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sbr, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module resourceid, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_TIME 1: cFE TIME Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 1: cFE TBL Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
1980-012-14:03:20.50224 CFE_ES_CreateObjects: Finished ES CreateObject table entries.
1980-012-14:03:20.50228 CFE_ES_Main: CFE_ES_Main entering CORE_READY state
1980-012-14:03:20.50231 CFE_ES_StartApplications: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.50261 CFE_ES_ParseFileEntry: Loading shared library: /cf/cfe_assert.so
1980-012-14:03:20.50282 [BEGIN] CFE FUNCTIONAL TEST
1980-012-14:03:20.50285 [BEGIN] 01 CFE-STARTUP
1980-012-14:03:20.50311 CFE_ES_ParseFileEntry: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.2.0-rc1+dev38, Last Official Release: v1.1.0
1980-012-14:03:20.50353 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.50412 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:03:20.50457 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.50499 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-14:03:20.55445 CFE_EVS_Register: Filter limit truncated to 8
1980-012-14:03:20.55456 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.4.0-rc1+dev42, Last Official Release: v2.3.0
1980-012-14:03:20.55485 CFE_EVS_Register: Filter limit truncated to 8
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable command.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
1980-012-14:03:20.60538 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.60542 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 66/1/TO_LAB_APP 3: TO telemetry output enabled for IP 127.0.0.1
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_ES 3: No-op command:
 cFS Versions: cfe v6.8.0-rc1+dev994, osal v5.1.0-rc1+dev604, psp v1.5.0-rc1+dev124
1980-012-14:43:51.50237 CFE_ES_RestartApp: Restart Application SCH_LAB_APP Initiated
1980-012-14:43:51.99992 CFE_ES_ExitApp: Called with invalid status (0).
1980-012-14:43:51.99994 CFE_ES_ExitApp: Application SCH_LAB_APP called CFE_ES_ExitApp
EVS Port1 66/1/CFE_ES 10: Restart Application SCH_LAB_APP Completed, AppID=1114122
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
EVS Port1 66/1/CFE_SB 28: No-op Cmd Rcvd:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
```

SB No-Op Command

Exercise 3 Recap



```
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1$ hexdump -C cf/map.bin
00000000  63 46 45 31 00 00 00 16  00 00 00 40 00 00 00 42  |cFE1.......@...B|
00000010  00 00 00 01 00 11 00 03  00 0f 55 47 00 ee 3c 8a  |..........UG..<.|
00000020  53 42 20 4d 61 70 20 49  6e 66 6f 72 6d 61 74 69  |SB Map Informati|
00000030  6f 6e 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |on..............|
00000040  01 18 00 00 00 00 00 00  09 18 00 00 01 00 00 00  |................|
00000050  03 18 00 00 02 00 00 00  0b 18 00 00 03 00 00 00  |................|
00000060  0e 18 00 00 04 00 00 00  08 18 00 00 05 00 00 00  |................|
00000070  06 18 00 00 06 00 00 00  0d 18 00 00 07 00 00 00  |................|
00000080  10 18 00 00 08 00 00 00  60 18 00 00 09 00 00 00  |........`.......|
00000090  11 18 00 00 0a 00 00 00  62 18 00 00 0b 00 00 00  |........b.......|
000000a0  05 18 00 00 0c 00 00 00  0c 18 00 00 0d 00 00 00  |................|
000000b0  04 18 00 00 0e 00 00 00  83 18 00 00 0f 00 00 00  |................|
000000c0  82 18 00 00 10 00 00 00  84 18 00 00 11 00 00 00  |................|
000000d0  85 18 00 00 12 00 00 00  80 18 00 00 13 00 00 00  |................|
000000e0  81 18 00 00 14 00 00 00  80 08 00 00 15 00 00 00  |................|
000000f0  81 08 00 00 16 00 00 00  84 08 00 00 17 00 00 00  |................|
00000100  83 08 00 00 18 00 00 00  00 08 00 00 19 00 00 00  |................|
00000110  01 08 00 00 1a 00 00 00  03 08 00 00 1b 00 00 00  |................|
00000120  04 08 00 00 1c 00 00 00  05 08 00 00 1d 00 00 00  |................|
00000130  06 08 00 00 1e 00 00 00  0a 08 00 00 1f 00 00 00  |................|
00000140  0c 08 00 00 20 00 00 00  08 08 00 00 21 00 00 00  |.... .......!...|
00000150  0b 08 00 00 22 00 00 00  10 08 00 00 23 00 00 00  |....".......#...|
00000160
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1$
```

🟩 File Header

🟨 Msg ID

🟥 Routing Table Index

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/133x0b1s.pdf

National Aeronautics and Space Administration

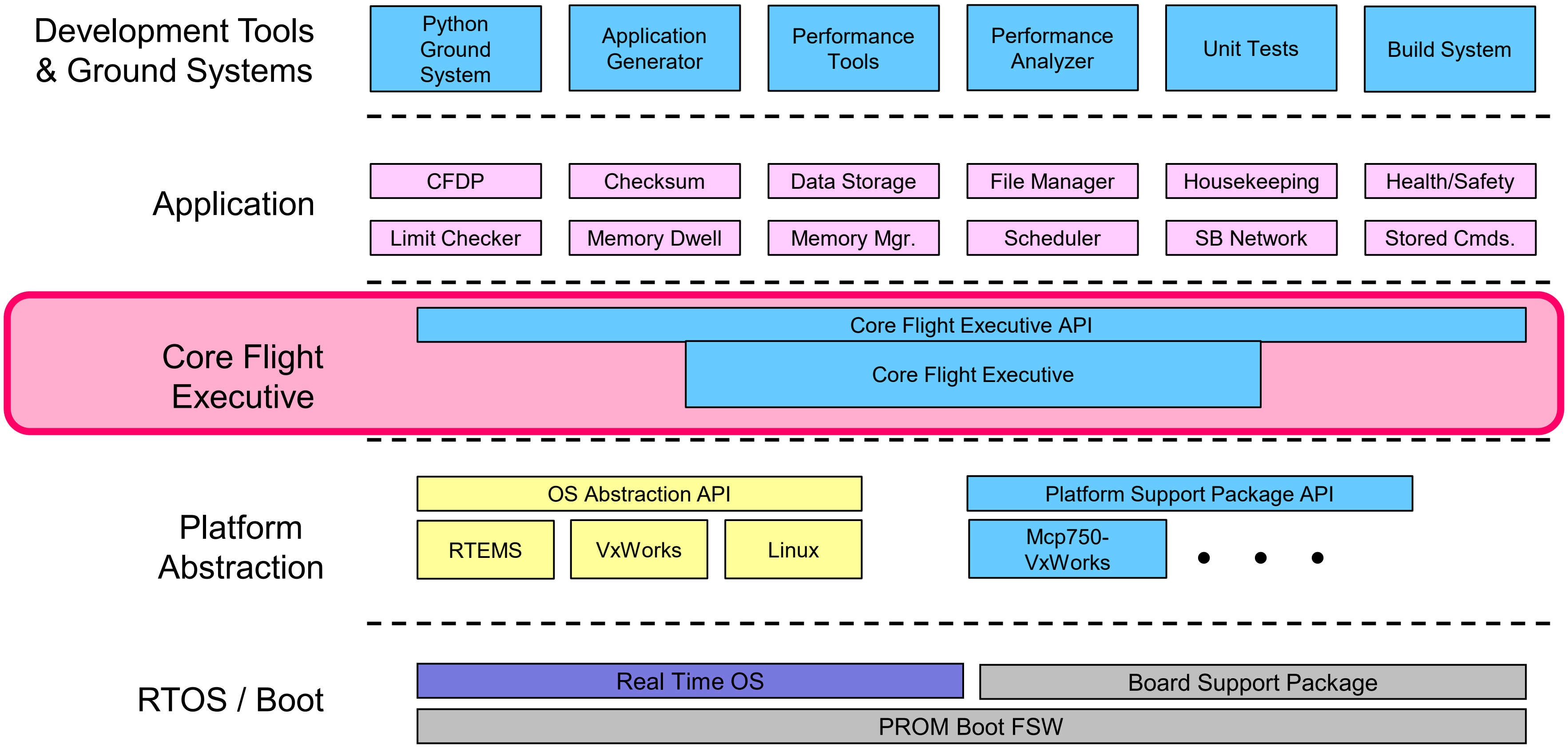
Core Flight System (cFS) Training

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 & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

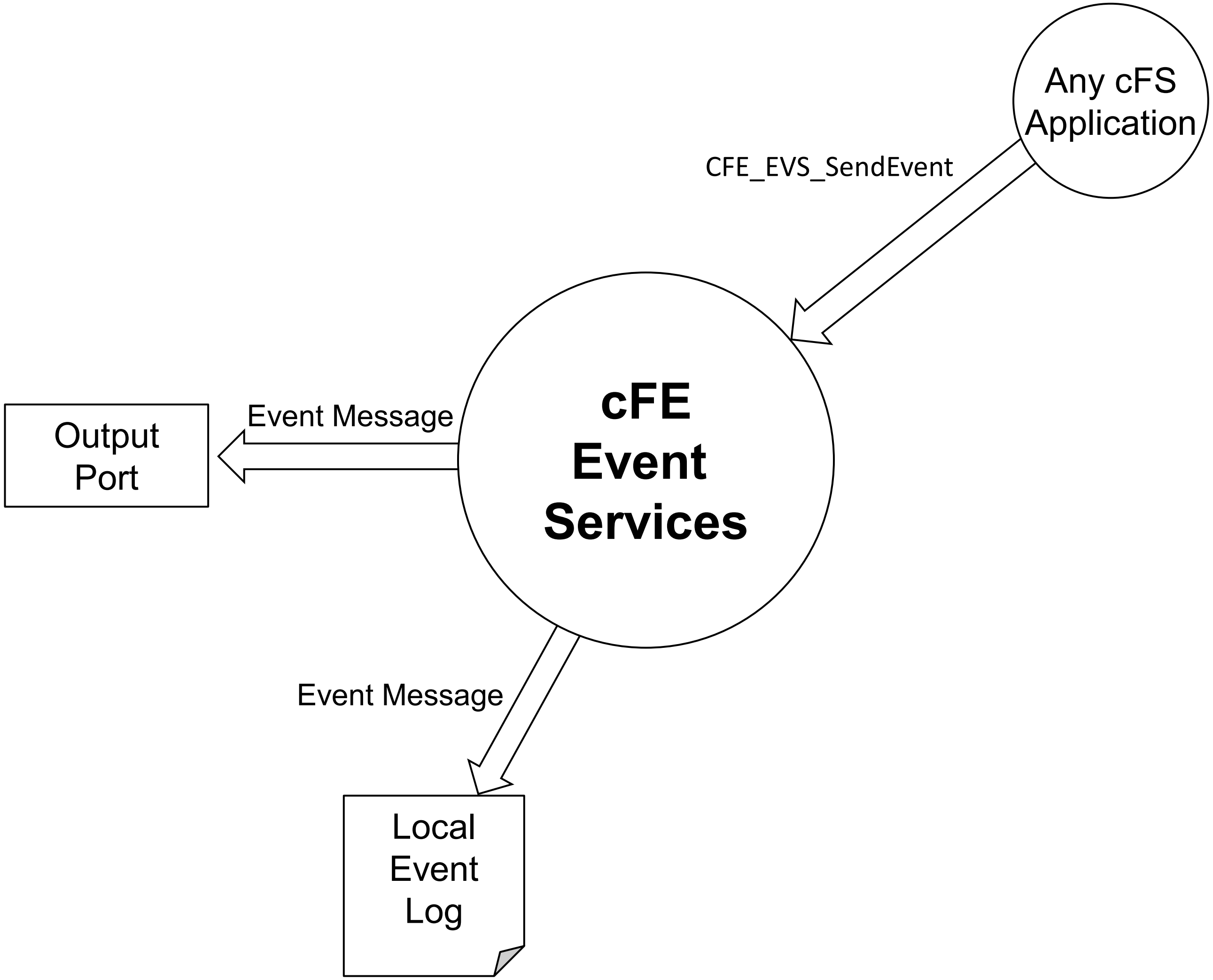
Event Services (EVS) - Overview

- **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 - 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 overwrite 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
  - Calls to any variety of `CFE_EVS_SendEvent` will have no effect if the app is not registered with EVS
  - cFE libraries cannot register with EVS

- **Event Filtering in Apps**
  - Apps should limit the amount of filtering done with in the app (ground should have ultimate control over filtering)
  - Apps should avoid “spamming” event messages

cFE Event Services APIs

Registration APIs	Purpose
CFE_EVS_Register	Register an application for receiving event services

Send Event APIs	Purpose
CFE_EVS_SendEvent	Generate a software event.
CFE_EVS_SendEventWithAppID	Generate a software event given the specified Application ID.
CFE_EVS_SendTimedEvent	Generate a software event with a specific time tag.

Reset Event Filter APIs	Purpose
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
CFE_EVS_SetLogModeCmd	Sets the logging mode to the command specified value.
CFE_EVS_WriteLogDataFileCmd	Requests the Event Service to generate a file containing the contents of the local event log.

Event Services – Platform Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_EVS_START_TASK_PRIORITY	Define EVS Task Priority
CFE_PLATFORM_EVS_START_TASK_STACK_SIZE	Define EVS Task Stack Size
CFE_PLATFORM_EVS_MAX_EVENT_FILTERS	Define Maximum Number of Event Filters per Application
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

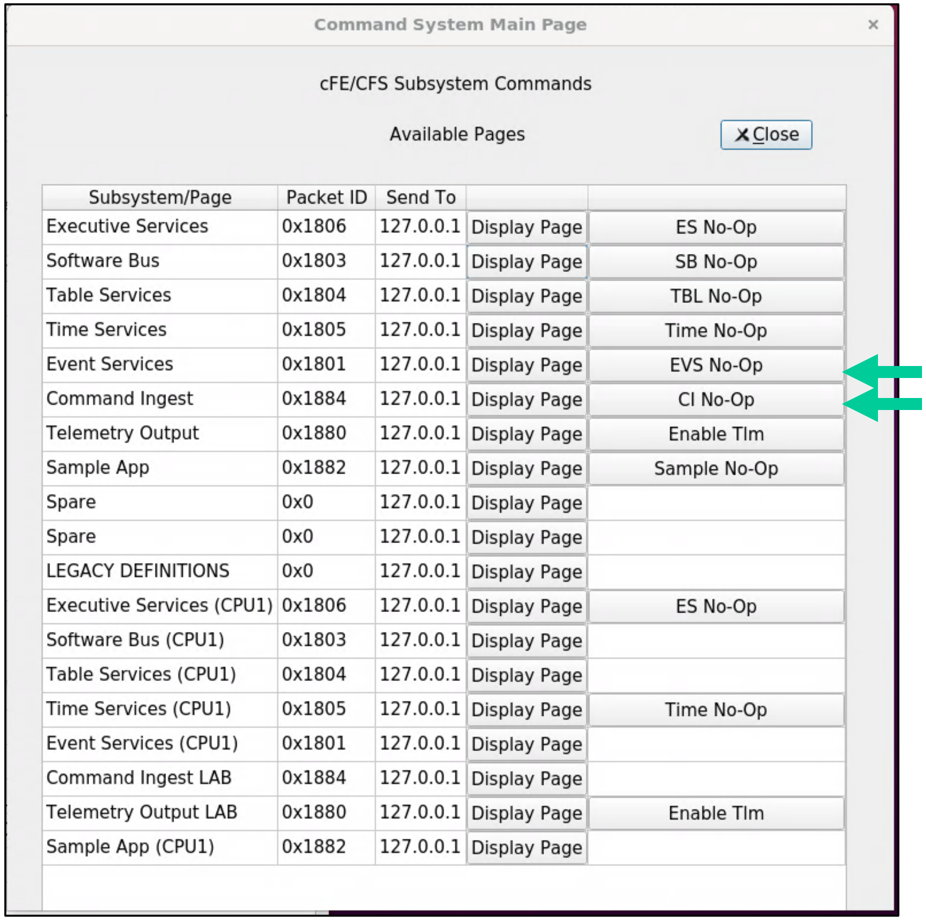
Event Services – Mission Configuration Parameters

Parameter	Purpose
CFE_MISSION_EVS_MAX_MESSAGE_LENGTH	Maximum Event Message Length

Exercise 4 - Command cFE Event Service

Part 1 – Test an Informational 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 “EVS No-Op“ button beside “Event Services”
6. Send a CI_LAB No-Op command
    - Click the “CI No-Op“ button beside “Command Ingest”



Command System Main Page

cFE/CFS Subsystem Commands

Available Pages

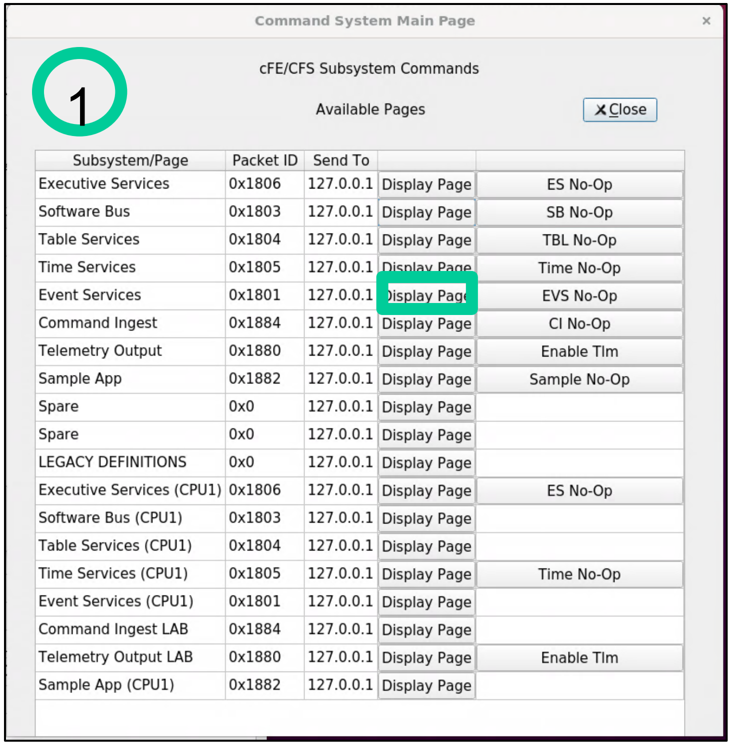
Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	
Spare	0x0	127.0.0.1	Display Page	
LEGACY DEFINITIONS	0x0	127.0.0.1	Display Page	
Executive Services (CPU1)	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus (CPU1)	0x1803	127.0.0.1	Display Page	
Table Services (CPU1)	0x1804	127.0.0.1	Display Page	
Time Services (CPU1)	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services (CPU1)	0x1801	127.0.0.1	Display Page	
Command Ingest LAB	0x1884	127.0.0.1	Display Page	
Telemetry Output LAB	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App (CPU1)	0x1882	127.0.0.1	Display Page	

Exercise 4 - Command cFE Event Service

Part 2 – Disable Informational Messages

1. Click the “Display Page” button beside “Event Services”
2. In the Event Services command window, click the "Send" button beside “CFE_EVS_DISABLE_EVENT_TYPE_CC"
3. Enter "2" as the "BitMask" Input and "0" as the "Spare" input.
4. Click send
5. Send a CI_LAB No-Op command
   - On the “Command System Main Page” window, click the “CI No-Op“ button beside “Command Ingest”

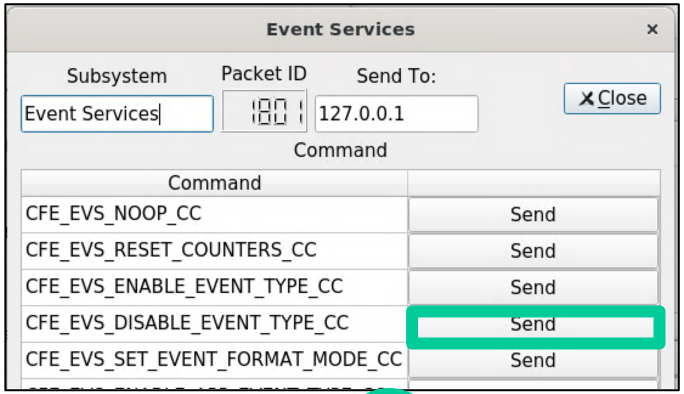
> Unlike the first time, nothing should show up in the cFE window. The CI_LAB no-op event message is an information level event message. Therefore, it was enabled until step #7 disabled informational messages.



cFE/CFS Subsystem Commands

Available Pages

Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	
Spare	0x0	127.0.0.1	Display Page	
LEGACY DEFINITIONS	0x0	127.0.0.1	Display Page	
Executive Services (CPU1)	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus (CPU1)	0x1803	127.0.0.1	Display Page	
Table Services (CPU1)	0x1804	127.0.0.1	Display Page	
Time Services (CPU1)	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services (CPU1)	0x1801	127.0.0.1	Display Page	
Command Ingest LAB	0x1884	127.0.0.1	Display Page	
Telemetry Output LAB	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App (CPU1)	0x1882	127.0.0.1	Display Page	



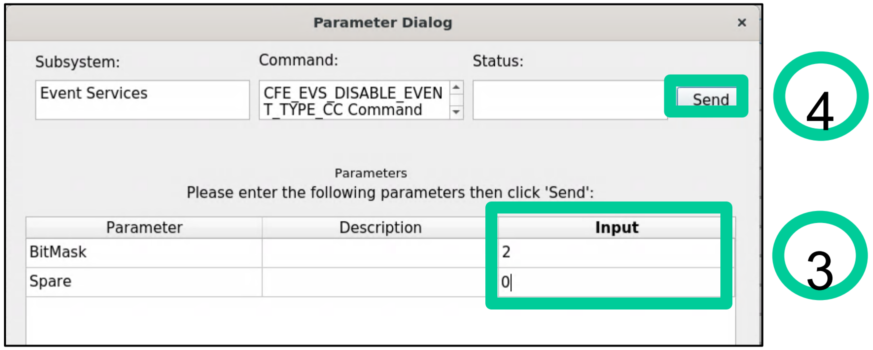
Subsystem: Event Services

Packet ID: 1801

Send To: 127.0.0.1

Command

Command	
CFE_EVS_NOOP_CC	Send
CFE_EVS_RESET_COUNTERS_CC	Send
CFE_EVS_ENABLE_EVENT_TYPE_CC	Send
CFE_EVS_DISABLE_EVENT_TYPE_CC	Send
CFE_EVS_SET_EVENT_FORMAT_MODE_CC	Send



Subsystem: Event Services

Command: CFE_EVS_DISABLE_EVENT_TYPE_CC Command

Status:

Parameters

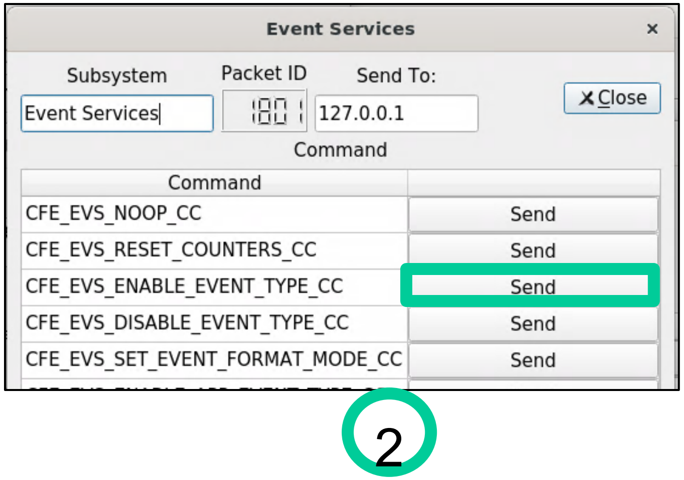
Please enter the following parameters then click 'Send':

Parameter	Description	Input
BitMask		2
Spare		0

Exercise 4 - Command cFE Event Service

[Optional] Re-enable informational messages

1. Click the “Display Page” button beside “Event Services”
2. In the Event Services command window, click the "Send" button beside “CFE_EVS_ENABLE_EVENT_TYPE_CC"
3. Enter "2" as the "BitMask" Input and "0" as the "Spare" input.
4. Click send

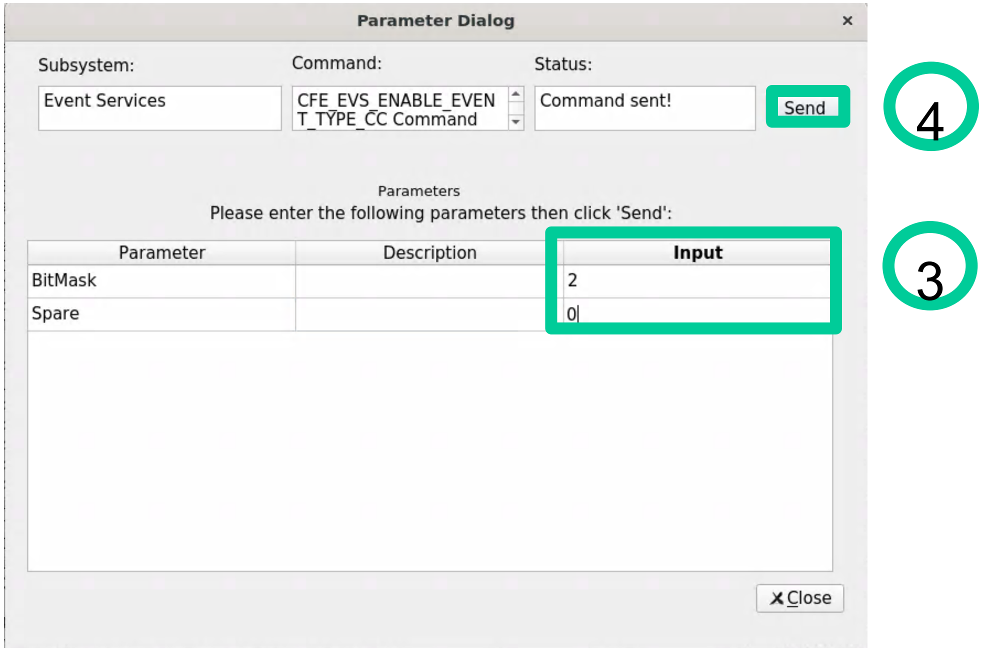


Event Services

Subsystem: Event Services
Packet ID: 180
Send To: 127.0.0.1

Command

Command	
CFE_EVS_NOOP_CC	Send
CFE_EVS_RESET_COUNTERS_CC	Send
CFE_EVS_ENABLE_EVENT_TYPE_CC	Send
CFE_EVS_DISABLE_EVENT_TYPE_CC	Send
CFE_EVS_SET_EVENT_FORMAT_MODE_CC	Send



Parameter Dialog

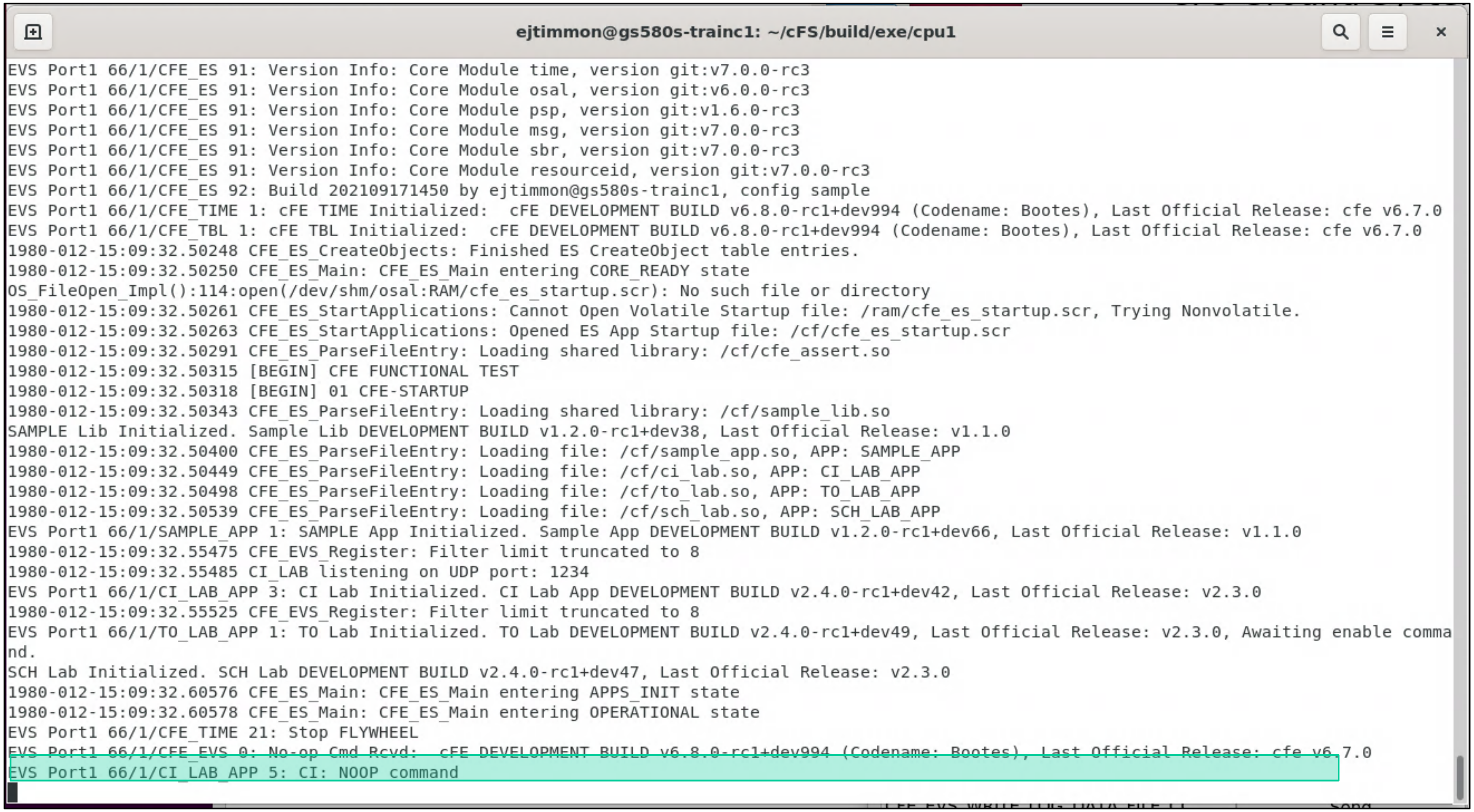
Subsystem: Event Services
Command: CFE_EVS_ENABLE_EVENT_TYPE_CC Command
Status: Command sent!

Parameters

Please enter the following parameters then click 'Send':

Parameter	Description	Input
BitMask		2
Spare		0

Exercise 4 Recap



```
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module time, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module osal, version git:v6.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module psp, version git:v1.6.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module msg, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sbr, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module resourceid, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_TIME 1: cFE TIME Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 1: cFE TBL Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
1980-012-15:09:32.50248 CFE_ES_CreateObjects: Finished ES CreateObject table entries.
1980-012-15:09:32.50250 CFE_ES_Main: CFE_ES_Main entering CORE_READY state
OS_FileOpen_Impl():114:open(/dev/shm/osal:RAM/cfe_es_startup.scr): No such file or directory
1980-012-15:09:32.50261 CFE_ES_StartApplications: Cannot Open Volatile Startup file: /ram/cfe_es_startup.scr, Trying Nonvolatile.
1980-012-15:09:32.50263 CFE_ES_StartApplications: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-15:09:32.50291 CFE_ES_ParseFileEntry: Loading shared library: /cf/cfe_assert.so
1980-012-15:09:32.50315 [BEGIN] CFE FUNCTIONAL TEST
1980-012-15:09:32.50318 [BEGIN] 01 CFE-STARTUP
1980-012-15:09:32.50343 CFE_ES_ParseFileEntry: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.2.0-rc1+dev38, Last Official Release: v1.1.0
1980-012-15:09:32.50400 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-15:09:32.50449 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-15:09:32.50498 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-15:09:32.50539 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-15:09:32.55475 CFE_EVS_Register: Filter limit truncated to 8
1980-012-15:09:32.55485 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.4.0-rc1+dev42, Last Official Release: v2.3.0
1980-012-15:09:32.55525 CFE_EVS_Register: Filter limit truncated to 8
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable comma
nd.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
1980-012-15:09:32.60576 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-15:09:32.60578 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 66/1/CFE_EVS 0: No-op Cmd Rcvd:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CI_LAB_APP 5: CI: NOOP command
```

CI No-Op Command

National Aeronautics and Space Administration

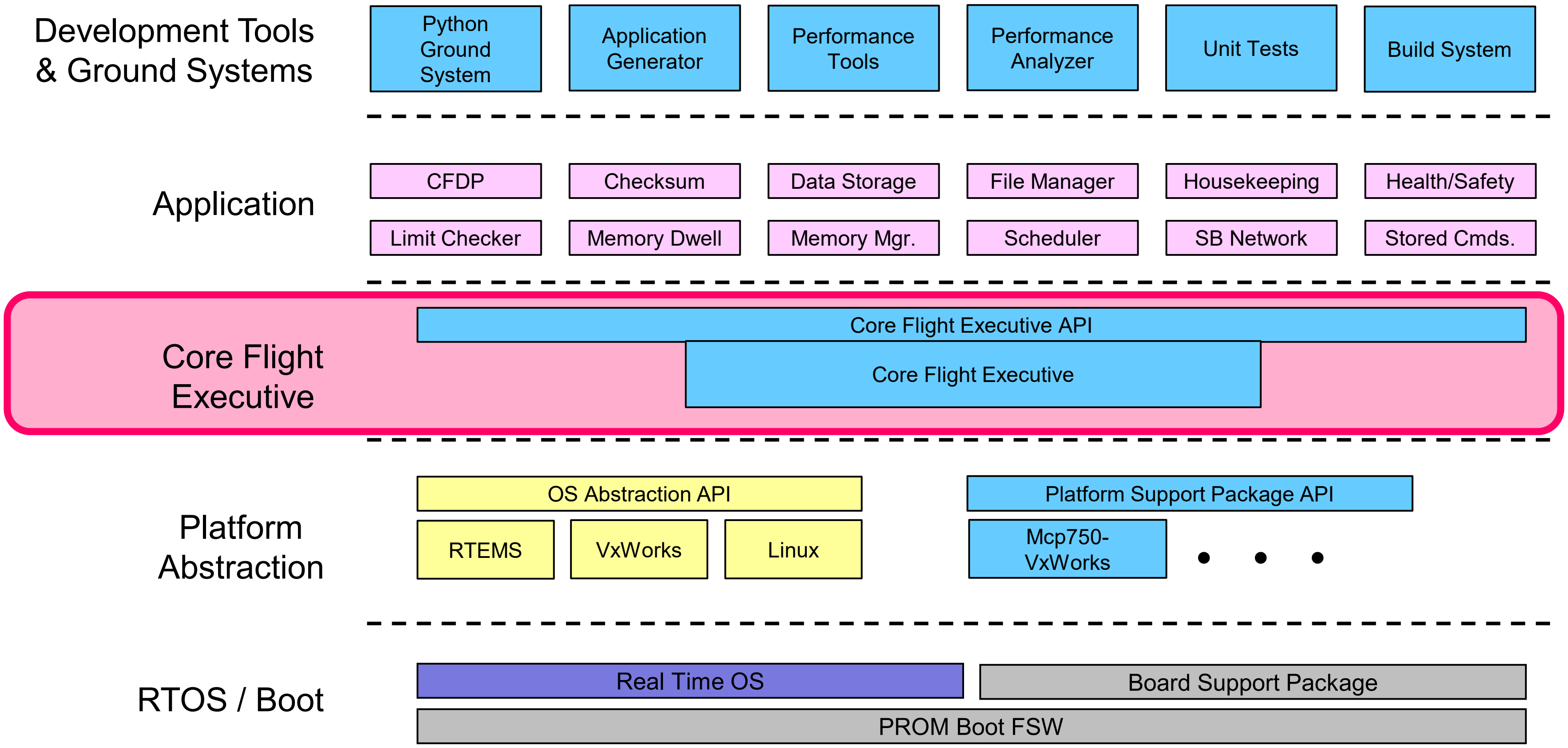
Core Flight System (cFS) Training

Module 2d: Time Services

Course Agenda

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

Time Services - cFS Context



Development Tools & Ground Systems

Python Ground System

Application Generator

Performance Tools

Performance Analyzer

Unit Tests

Build System

Application

CFDP

Checksum

Data Storage

File Manager

Housekeeping

Health/Safety

Limit Checker

Memory Dwell

Memory Mgr.

Scheduler

SB Network

Stored Cmds.

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

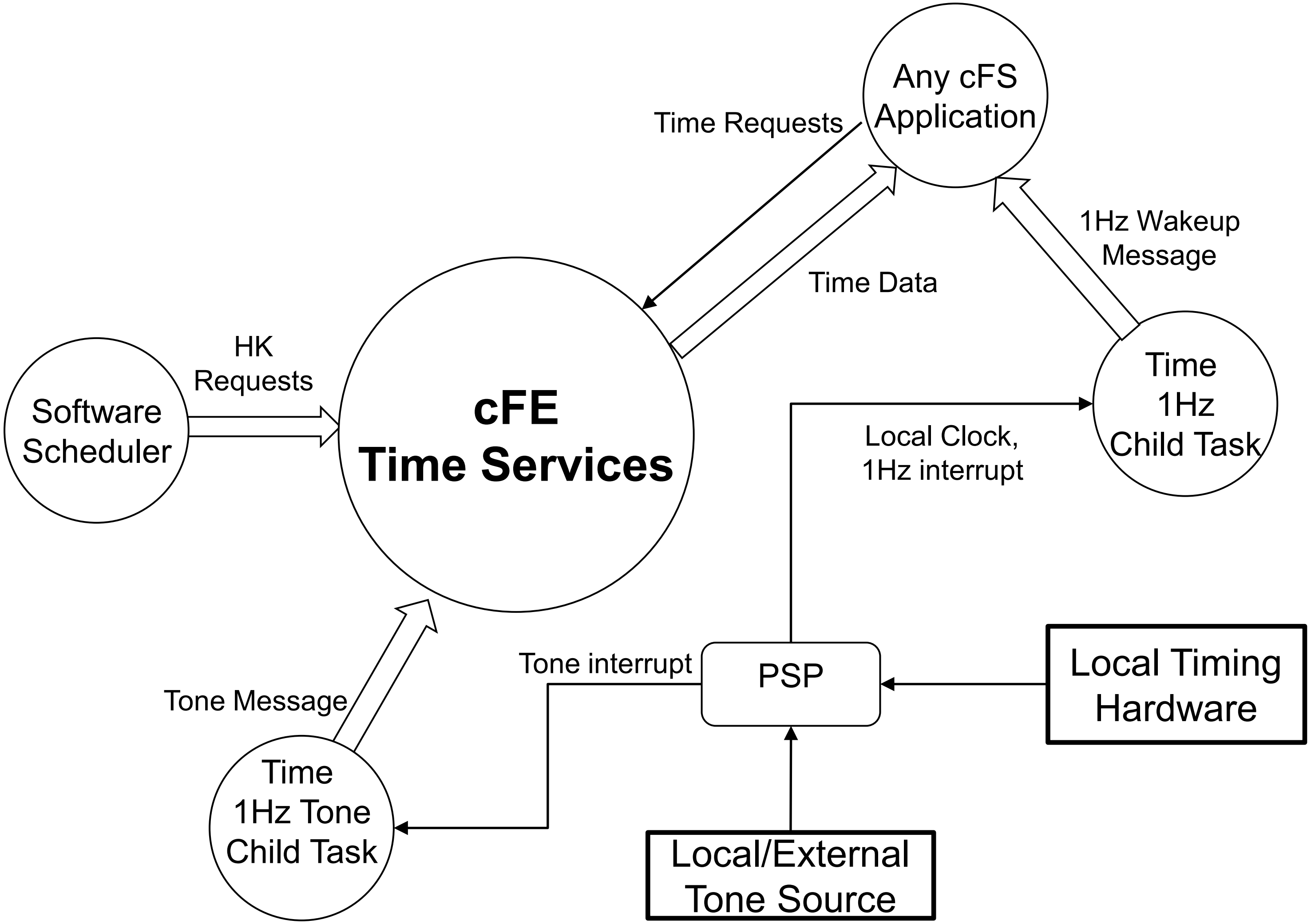
Time Services - Overview

- 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 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

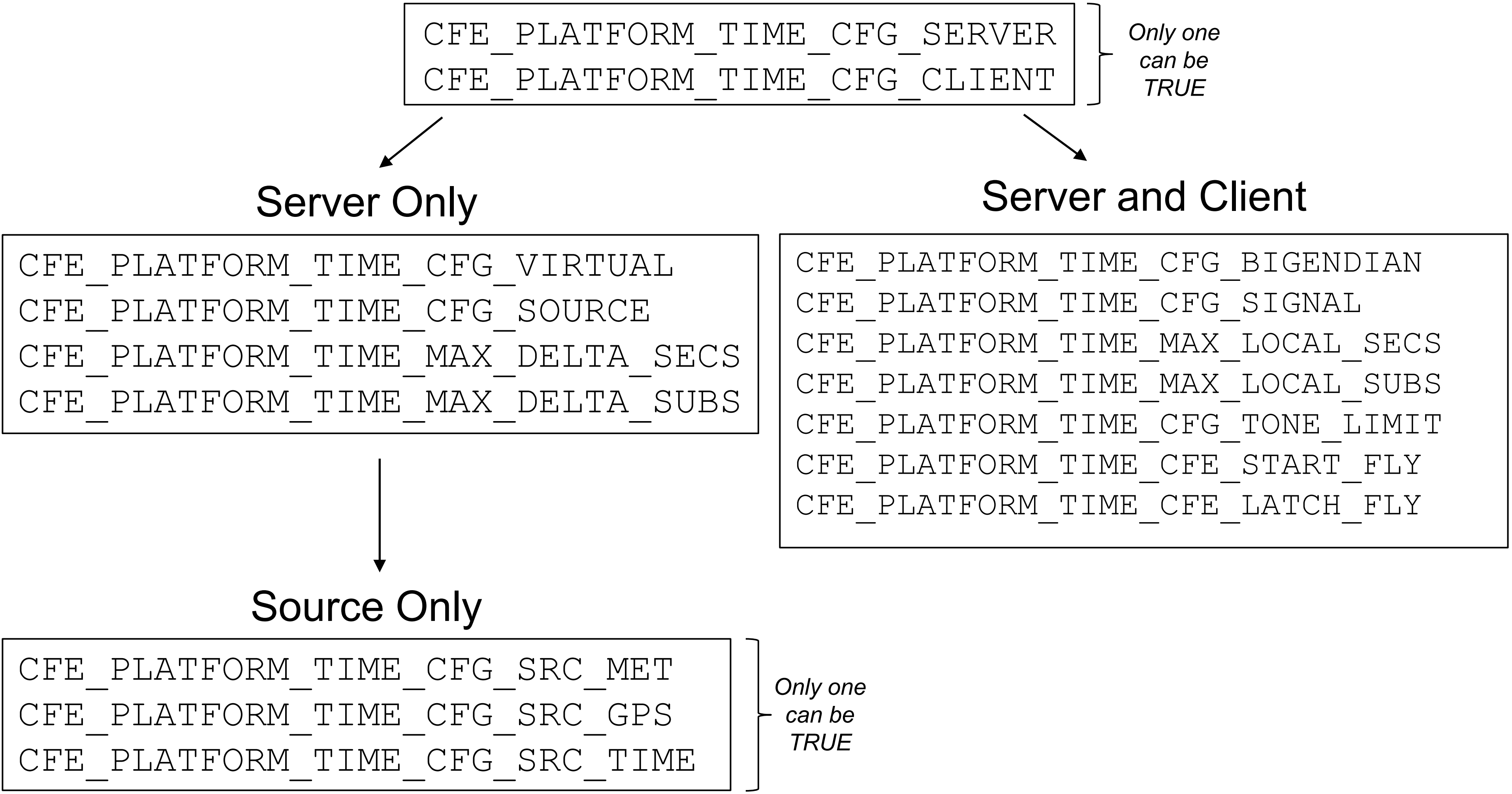
Time Services – Retrieving Onboard State

- **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

cFE Time Services APIs

Get Current Time APIs	Purpose
CFE_TIME_GetTime	Get the current spacecraft time
CFE_TIME_GetTAI	Get the current TAI (MET + SCTF) time
CFE_TIME_GetUTC	Get the current UTC (MET + SCTF - Leap Seconds) time
CFE_TIME_GetMET	Get the current value of the Mission Elapsed Time (MET)
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

Get Time Information APIs	Purpose
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	Provides information about the spacecraft clock

Time Arithmetic APIs	Purpose
CFE_TIME_Add	Adds two time values
CFE_TIME_Subtract	Subtracts two time values
CFE_TIME_Compare	Compares two time values

cFE Time Services APIs

Time Conversion APIs	Purpose
CFE_TIME_MET2SCTime	Convert specified MET into Spacecraft Time
CFE_TIME_Sub2MicroSecs	Converts a sub-seconds count to an equivalent number of microseconds
CFE_TIME_Micro2SubSecs	Converts a number of microseconds to an equivalent sub-seconds count

External Time Source APIs	Purpose
CFE_TIME_ExternalTone	Provides the 1 Hz signal from an external source
CFE_TIME_ExternalMET	Provides the Mission Elapsed Time from an external source
CFE_TIME_ExternalGPS	Provide the time from an external source that has data common to GPS receivers
CFE_TIME_ExternalTime	Provide the time from an external source that measures time relative to a known epoch
CFE_TIME_RegisterSynchCallback	Registers a callback function that is called whenever time synchronization occurs
CFE_TIME_UnregisterSynchCallback	Unregisters a callback function that is called whenever time synchronization occurs

Miscellaneous Time APIs	Purpose
CFE_TIME_Print	Print a time value as a string
CFE_TIME_Local1HzISR	Drives the time processing logic from the system PSP layer.

Time Services Commands

Command Functions	Purpose
CFE_TIME_Add1HZAdjustmentCmd	Add Delta to Spacecraft Time Correlation Factor each 1Hz
CFE_TIME_AddAdjustCmd	Add Delta to Spacecraft Time Correlation Factor
CFE_TIME_AddDelayCmd	Add Time to Tone Time Delay
CFE_TIME_SendDiagnosticTlm	Request TIME Diagnostic Telemetry
CFE_TIME_NoopCmd	Time No-Op
CFE_TIME_ResetCountersCmd	Resets counters within the housekeeping telemetry
CFE_TIME_SetLeapSecondsCmd	Set Leap Seconds
CFE_TIME_SetMETCmd	Set Mission Elapsed Time
CFE_TIME_SetSignalCmd	Set Tone Signal Source
CFE_TIME_SetSourceCmd	Set Time Source
CFE_TIME_SetStateCmd	Set Time State
CFE_TIME_SetSTCFCmd	Set Spacecraft Time Correlation Factor
CFE_TIME_SetTimeCmd	Set Spacecraft Time
CFE_TIME_Sub1HZAdjustmentCmd	Subtract Delta from Spacecraft Time Correlation Factor each 1Hz
CFE_TIME_SubAdjustCmd	Subtract Delta from Spacecraft Time Correlation Factor
CFE_TIME_SubDelayCmd	Subtract Time from Tone Time Delay

Time Services – Platform Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_TIME_CFG_[SERVER/CLIENT]	Time Server or Time Client Selection
CFE_PLATFORM_TIME_CFG_BIGENDIAN	Time Tone In Big-Endian Order
CFE_PLATFORM_TIME_CFG_VIRTUAL	Local MET or Virtual MET Selection for Time Servers
CFE_PLATFORM_TIME_CFG_SIGNAL	Include or Exclude the Primary/Redundant Tone Selection Cmd
CFE_PLATFORM_TIME_CFG_SOURCE	Include or Exclude the Internal/External Time Source Selection Cmd
CFE_PLATFORM_TIME_CFG_SRC_[MET/GPS/TIME]	Choose the External Time Source for Server only
CFE_PLATFORM_TIME_MAX_DELTA_[SECS/SUBS]	Define the Max Delta Limits for Time Servers using an Ext Time Source
CFE_PLATFORM_TIME_MAX_LOCAL_[SECS/SUBS]	Define the Local Clock Rollover Value in seconds and subseconds
CFE_PLATFORM_TIME_CFG_TONE_LIMIT	Define Timing Limits From One Tone To The Next
CFE_PLATFORM_TIME_CFG_START_FLY	Define Time to Start Flywheel Since Last Tone
CFE_PLATFORM_TIME_CFG_LATCH_FLY	Define Periodic Time to Update Local Clock Tone Latch
CFE_PLATFORM_TIME_START_TASK_PRIORITY	Defines the cFE_TIME Task priority.
CFE_PLATFORM_TIME_TONE_TASK_PRIORITY	Defines the cFE_TIME Tone Task priority.
CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY	Defines the cFE_TIME 1HZ Task priority.

Time Services – Platform Configuration Parameters

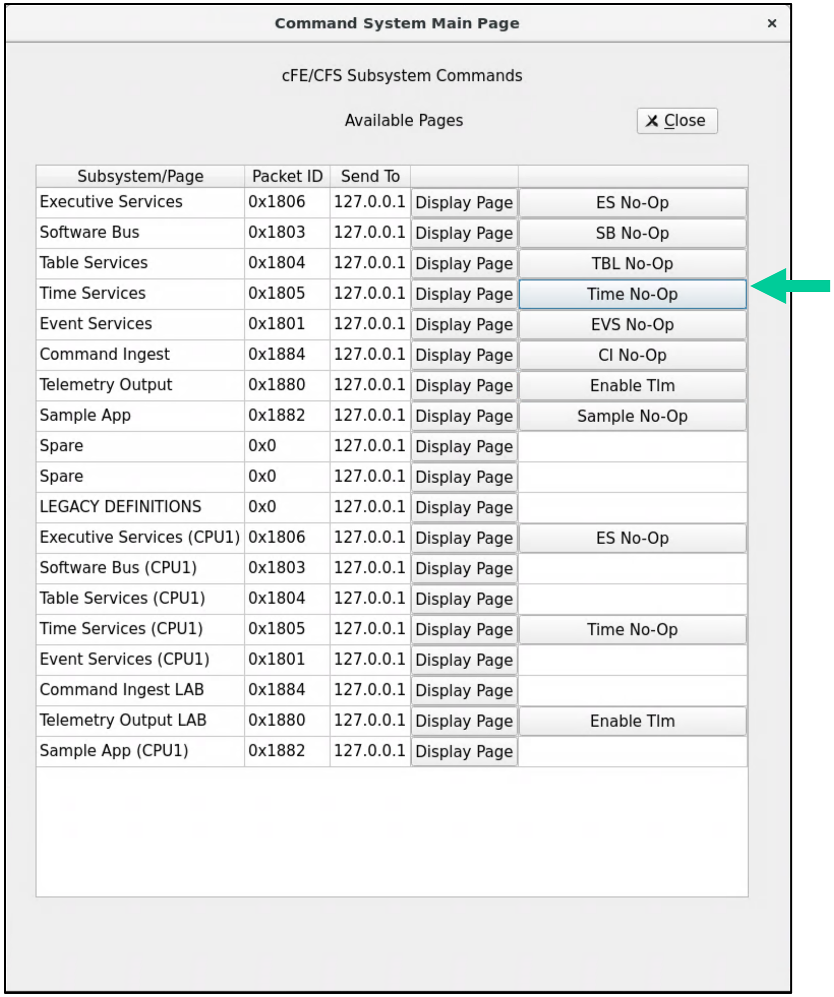
Parameter	Purpose
CFE_PLATFORM_TIME_START_TASK_STACK_SIZE	Defines the cFE_TIME Main Task Stack Size
CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE	Defines the cFE_TIME Tone Task Stack Size
CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE	Defines the cFE_TIME 1HZ Task Stack Size

Time Services – Mission Configuration Parameters

Parameter	Purpose
CFE_MISSION_TIME_CFG_DEFAULT_[TAI/UTC]	Select either UTC or TAI as the default (mission specific) time format.
CFE_MISSION_TIME_CFG_FAKE_TONE	Default Time Format
CFE_MISSION_TIME_AT_TONE_[WAS/WILL_BE]	Default Time and Tone Order
CFE_MISSION_TIME_MIN_ELAPSED	Min Time Elapsed
CFE_MISSION_TIME_MAX_ELAPSED	Max Time Elapsed
CFE_MISSION_TIME_DEF_MET_[SECS/SUBS]	Default Time Values
CFE_MISSION_TIME_DEF_STCF_[SECS/SUBS]	Default Time Values
CFE_MISSION_TIME_DEF_DELAY_[SECS/SUBS]	Default Time Values
CFE_MISSION_TIME_DEF_LEAPS	Default Time Values
CFE_MISSION_TIME_EPOCH_YEAR	Default ground time epoch values
CFE_MISSION_TIME_EPOCH_DAY	Default ground time epoch values
CFE_MISSION_TIME_EPOCH_HOUR	Default ground time epoch values
CFE_MISSION_TIME_EPOCH_MINUTE	Default ground time epoch values
CFE_MISSION_TIME_EPOCH_SECOND	Default ground time epoch values
CFE_MISSION_TIME_FS_FACTOR	Define the s/c vs file system time conversion constant

Exercise 5 - 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. Enable Telemetry (as in Exercise 2)
5. Send a TIME No-Op command
   - Click the “Time No-Op“ button beside “Time Services”



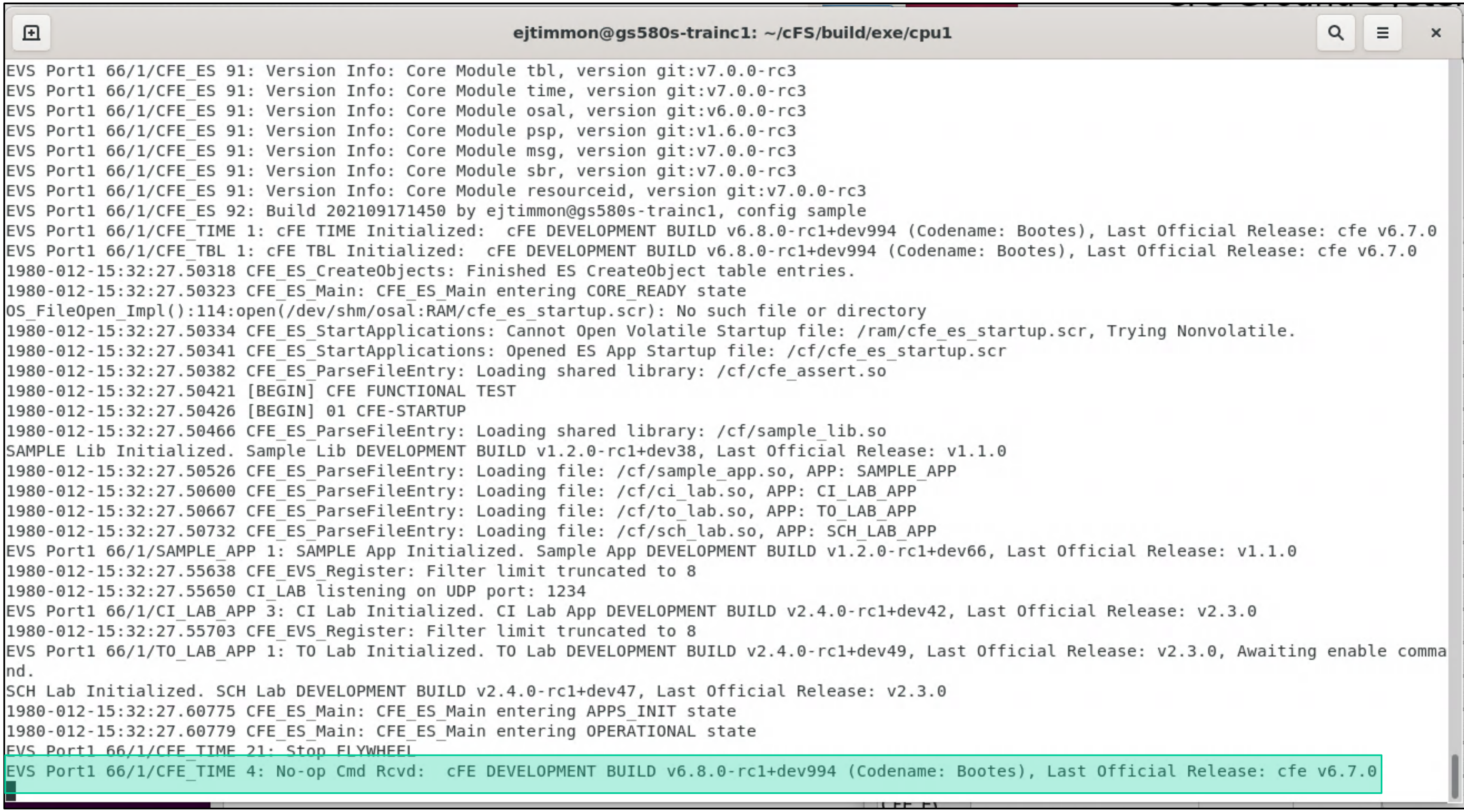
Command System Main Page

cFE/CFS Subsystem Commands

Available Pages

Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0.1	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	
Spare	0x0	127.0.0.1	Display Page	
LEGACY DEFINITIONS	0x0	127.0.0.1	Display Page	
Executive Services (CPU1)	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus (CPU1)	0x1803	127.0.0.1	Display Page	
Table Services (CPU1)	0x1804	127.0.0.1	Display Page	
Time Services (CPU1)	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services (CPU1)	0x1801	127.0.0.1	Display Page	
Command Ingest LAB	0x1884	127.0.0.1	Display Page	
Telemetry Output LAB	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App (CPU1)	0x1882	127.0.0.1	Display Page	

Exercise 5 Recap



```
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module tbl, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module time, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module osal, version git:v6.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module psp, version git:v1.6.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module msg, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sbr, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module resourceid, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_TIME 1: cFE TIME Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 1: cFE TBL Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
1980-012-15:32:27.50318 CFE_ES_CreateObjects: Finished ES CreateObject table entries.
1980-012-15:32:27.50323 CFE_ES_Main: CFE_ES_Main entering CORE_READY state
OS_FileOpen_Impl():114:open(/dev/shm/osal:RAM/cfe_es_startup.scr): No such file or directory
1980-012-15:32:27.50334 CFE_ES_StartApplications: Cannot Open Volatile Startup file: /ram/cfe_es_startup.scr, Trying Nonvolatile.
1980-012-15:32:27.50341 CFE_ES_StartApplications: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-15:32:27.50382 CFE_ES_ParseFileEntry: Loading shared library: /cf/cfe_assert.so
1980-012-15:32:27.50421 [BEGIN] CFE FUNCTIONAL TEST
1980-012-15:32:27.50426 [BEGIN] 01 CFE-STARTUP
1980-012-15:32:27.50466 CFE_ES_ParseFileEntry: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.2.0-rc1+dev38, Last Official Release: v1.1.0
1980-012-15:32:27.50526 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-15:32:27.50600 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-15:32:27.50667 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-15:32:27.50732 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-15:32:27.55638 CFE_EVS_Register: Filter limit truncated to 8
1980-012-15:32:27.55650 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.4.0-rc1+dev42, Last Official Release: v2.3.0
1980-012-15:32:27.55703 CFE_EVS_Register: Filter limit truncated to 8
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable comma
nd.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
1980-012-15:32:27.60775 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-15:32:27.60779 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 66/1/CFE_TIME 4: No-op Cmd Rcvd:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
```

TIME
No-Op
Command

National Aeronautics and Space Administration

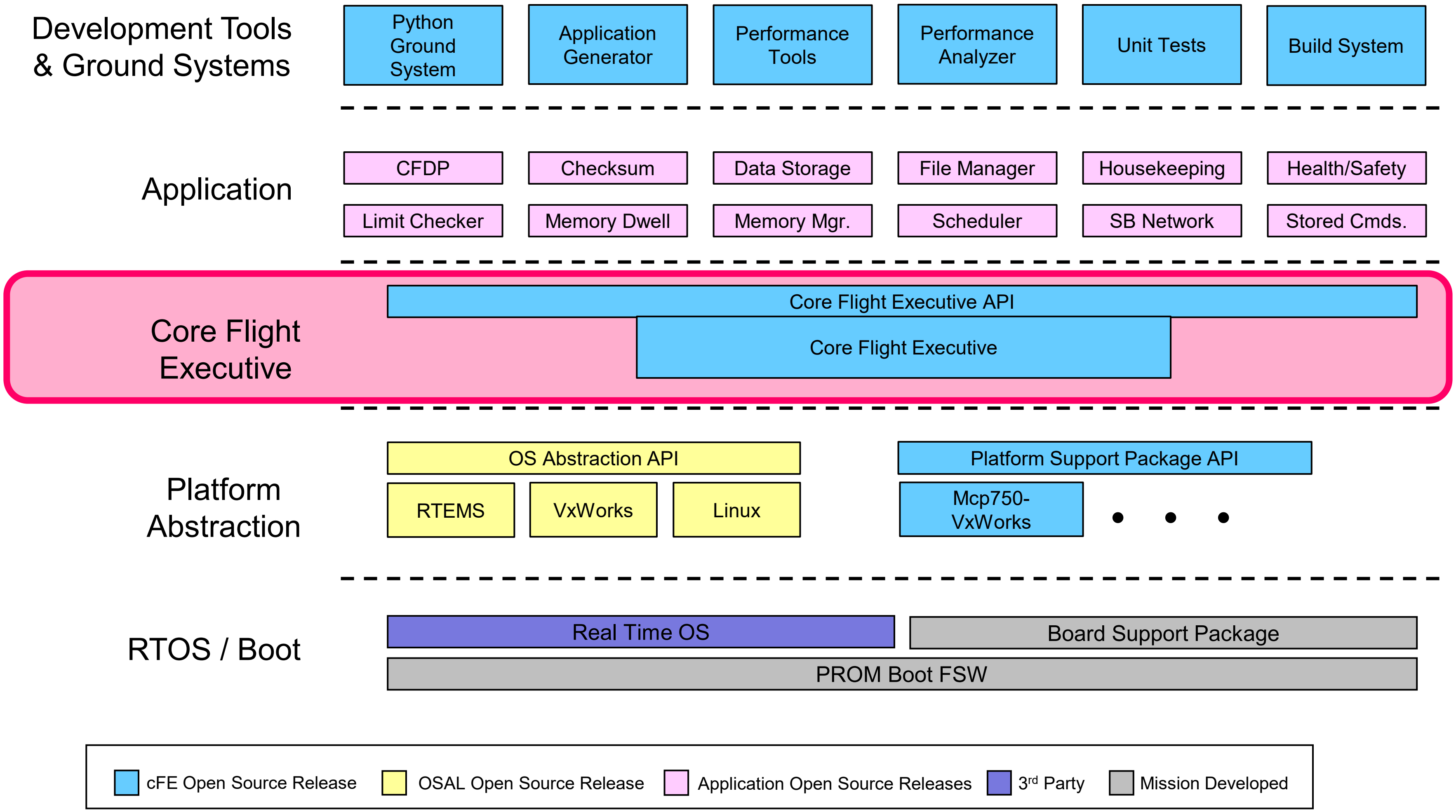
Core Flight System (cFS) Training

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 & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

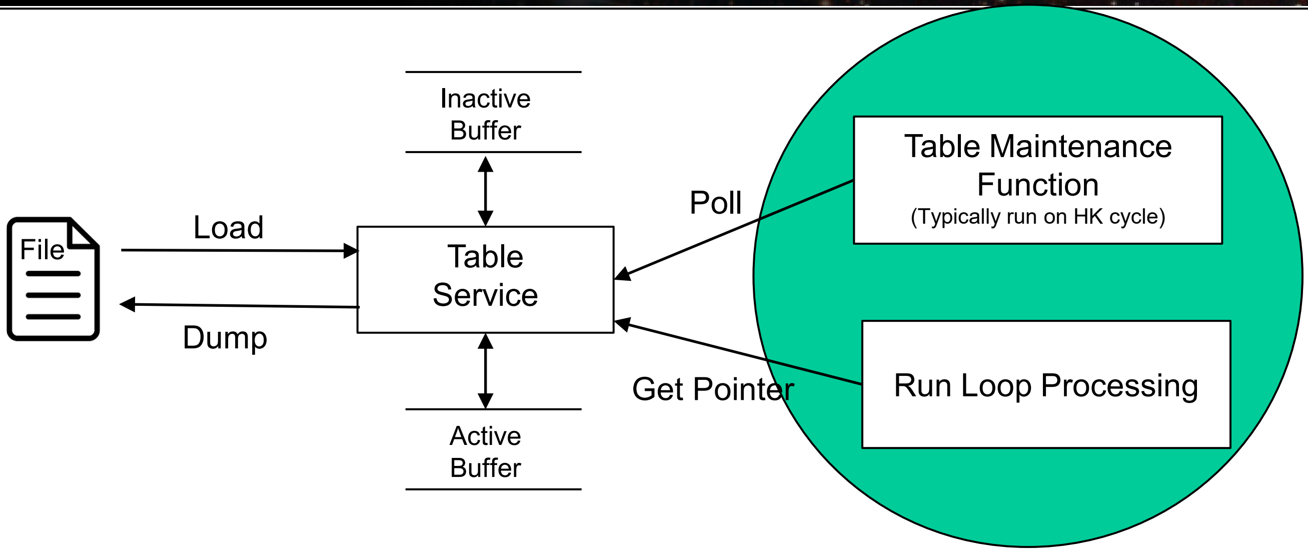
PROM Boot FSW

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

Table Services (TBL) - Overview

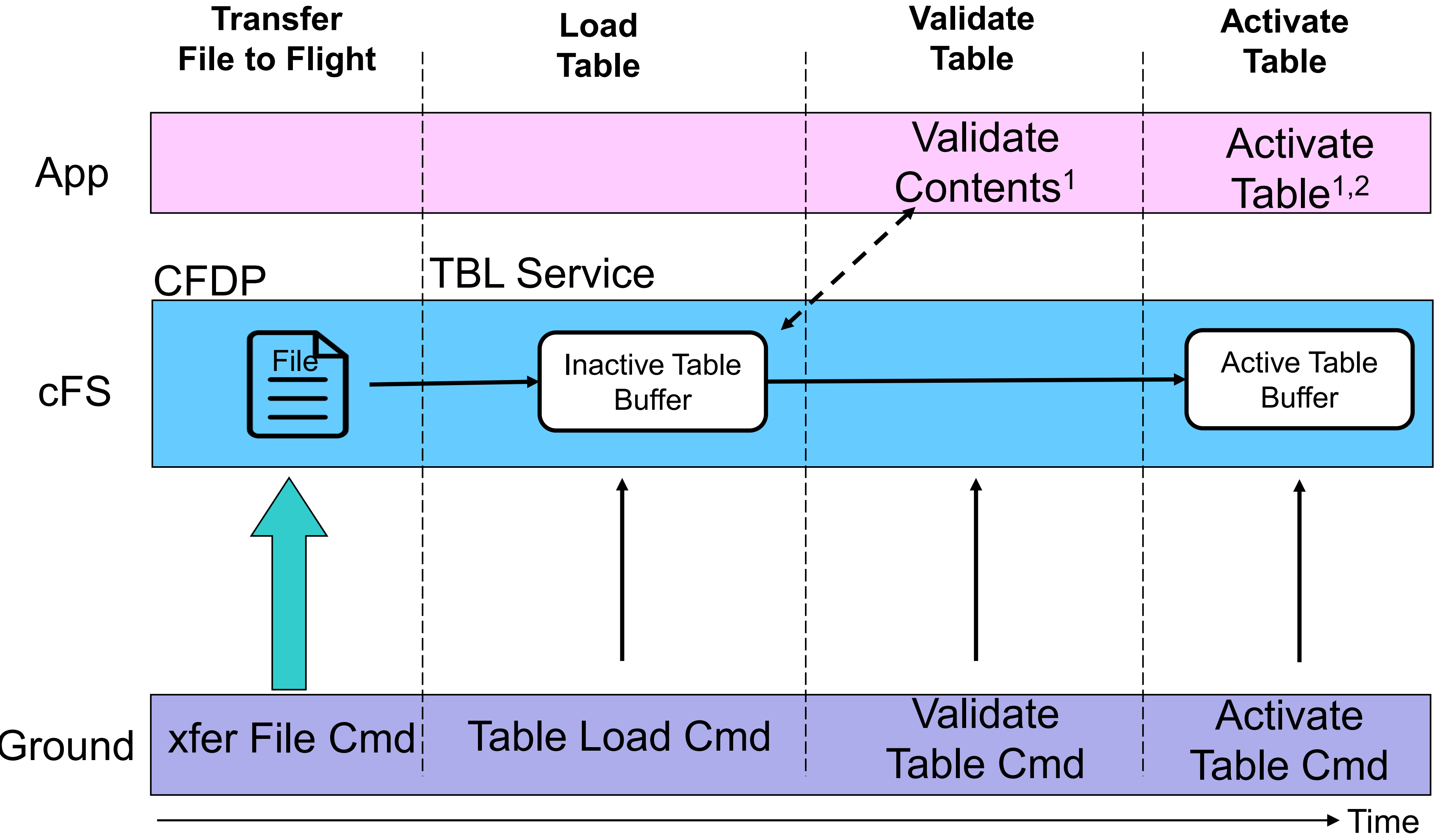
- **What is a table?**
  - Tables are logical groups of parameters that are managed as a named entity
- **Parameters typically change the behavior of a FSW algorithm**
  - Examples include controller gains, conversion factors, and filter algorithm parameters
- **Tables service provides ground commands 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

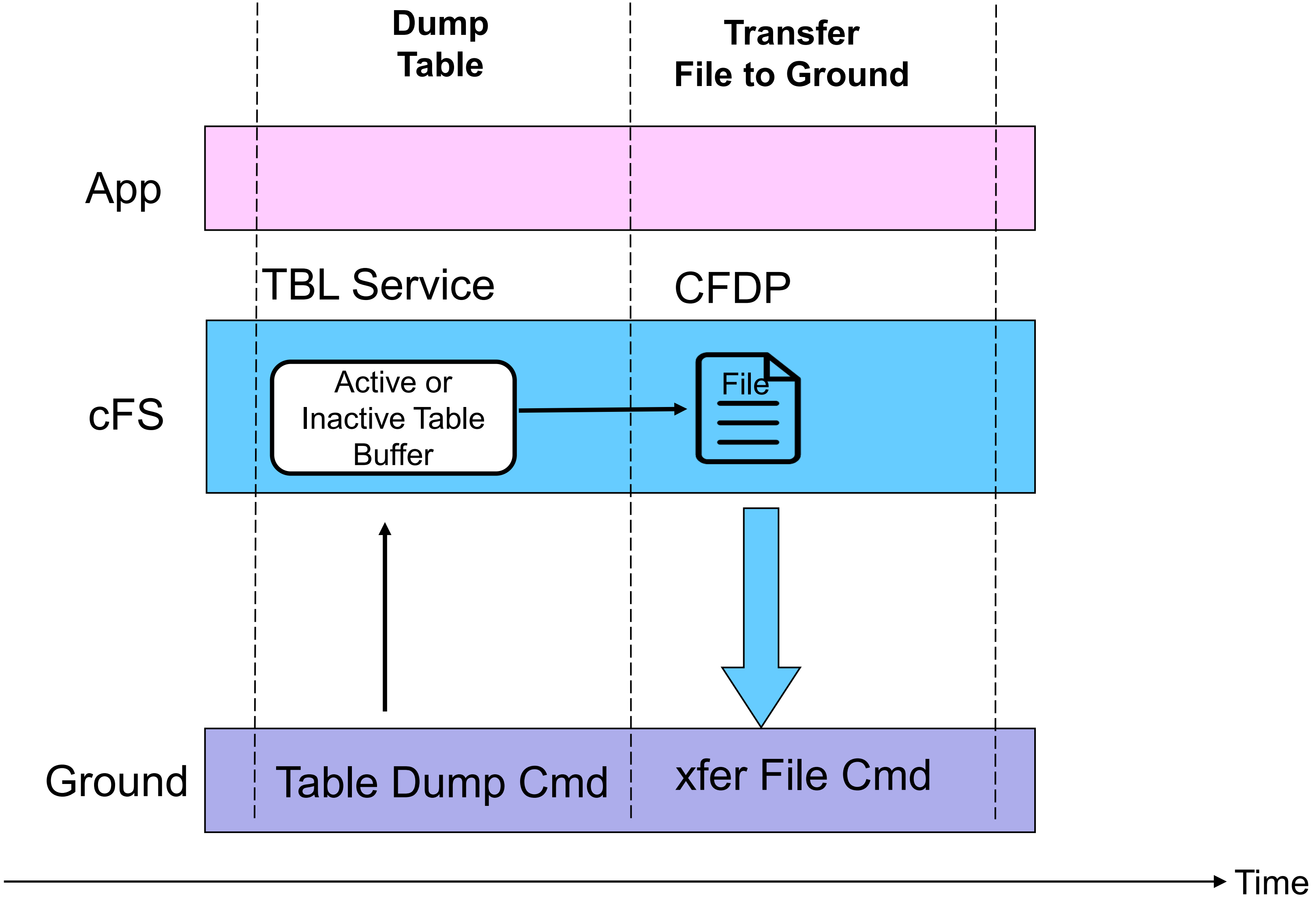


Table Services –Table Buffer Types

- **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 dumped
  - Last table loaded

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

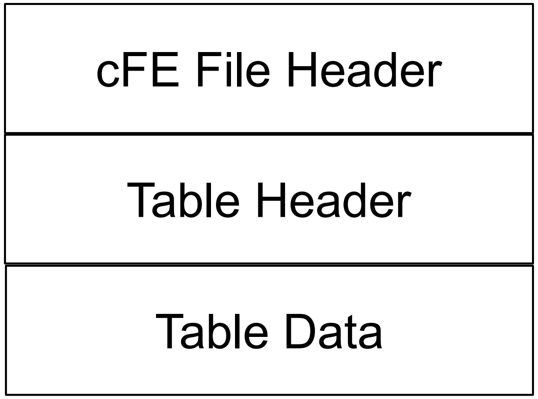
- **Dump Table Registry**
  - Write the pertinent table registry information to the command-specified file

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;
```

cFE Table Services APIs

Registration APIs	Purpose
CFE_TBL_Register	Register a table with cFE to obtain Table Management Services
CFE_TBL_Share	Obtain handle of table registered by another application
CFE_TBL_Unregister	Unregister a table

Manage Table Content APIs	Purpose
CFE_TBL_Load	Load a specified table with data from specified source
CFE_TBL_Update	Update contents of a specified table, if an update is pending
CFE_TBL_Validate	Perform steps to validate the contents of a table image
CFE_TBL_Manage	Perform standard operations to maintain a table
CFE_TBL_DumpToBuffer	Copies the contents of a Dump Only Table to a shared buffer
CFE_TBL_Modified	Notify cFE Table Services that table contents have been modified by the Application

Access Table Content APIs	Purpose
CFE_TBL_GetAddress	Obtain the current address of the contents of the specified table
CFE_TBL_ReleaseAddress	Release previously obtained pointer to the contents of the specified table
CFE_TBL_GetAddresses	Obtain the current addresses of an array of specified tables
CFE_TBL_ReleaseAddresses	Release the addresses of an array of specified tables

cFE Table Services APIs

Get Table Information APIs	Purpose
CFE_TBL_GetStatus	Obtain current status of pending actions for a table
CFE_TBL_GetInfo	Obtain characteristics/information of/about a specified table
CFE_TBL_NotifyByMessage	Instruct cFE Table Services to notify Application via message when table requires management

Table Services Commands

Command Functions	Purpose
CFE_TBL_NoopCmd	Table No-Op
CFE_TBL_ResetCountersCmd	Resets the counters within the Table Services housekeeping telemetry
CFE_TBL_LoadCmd	Loads the contents of the specified file into an inactive buffer for the table specified within the file.
CFE_TBL_DumpCmd	This command will cause the Table Services to put the contents of the specified table buffer into the command specified file.
CFE_TBL_ValidateCmd	Validate Table
CFE_TBL_ActivateCmd	Activate Table
CFE_TBL_DumpRegistryCmd	This command will cause Table Services to write some of the contents of the Table Registry to the command specified file.
CFE_TBL_SendRegistryCmd	This command will cause Table Services to telemeter the contents of the Table Registry for the command specified table.
CFE_TBL_DeleteCDSCmd	This command will delete the Critical Data Store (CDS) associated with the specified Critical Table.
CFE_TBL_AbortLoadCmd	This command will cause Table Services to discard the contents of a table buffer that was previously loaded with the data in a file as specified by a Table Load command.

Table Services – Platform Configuration Parameters

Parameter	Purpose
CFE_PLATFORM_TBL_START_TASK_PRIORITY	Defines the cFE_TBL Task priority
CFE_PLATFORM_TBL_START_TASK_STACK_SIZE	Define TBL Task Stack Size
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_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_[1/2/3/4]	Processor ID values used for table load validation

Table Services – Mission Configuration Parameters

Parameter	Purpose
CFE_MISSION_TBL_MAX_NAME_LENGTH	Maximum Table Name Length
CFE_MISSION_TBL_MAX_FULL_NAME_LEN	Maximum Length of Full Table Name in messages

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 a TBL No-Op command
    - Click the “TBL No-Op“ button beside “Table Services”
6. Send a "Load Table" command
    - Click the “Display Page” button beside “Table Services”
    - In the "Table Services" window, click the "Send" button beside “CFE_TBL_LOAD_CC"
    - Enter “/cf/sample_app_tbl.tbl" in the "Input" field next to "LoadFilename"
    - Click "Send"
7. Dump the table registry
    - In the "Table Services " window, click the "Send" button beside “CFE_TBL_DUMP_REGISTRY_CC"
    - 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

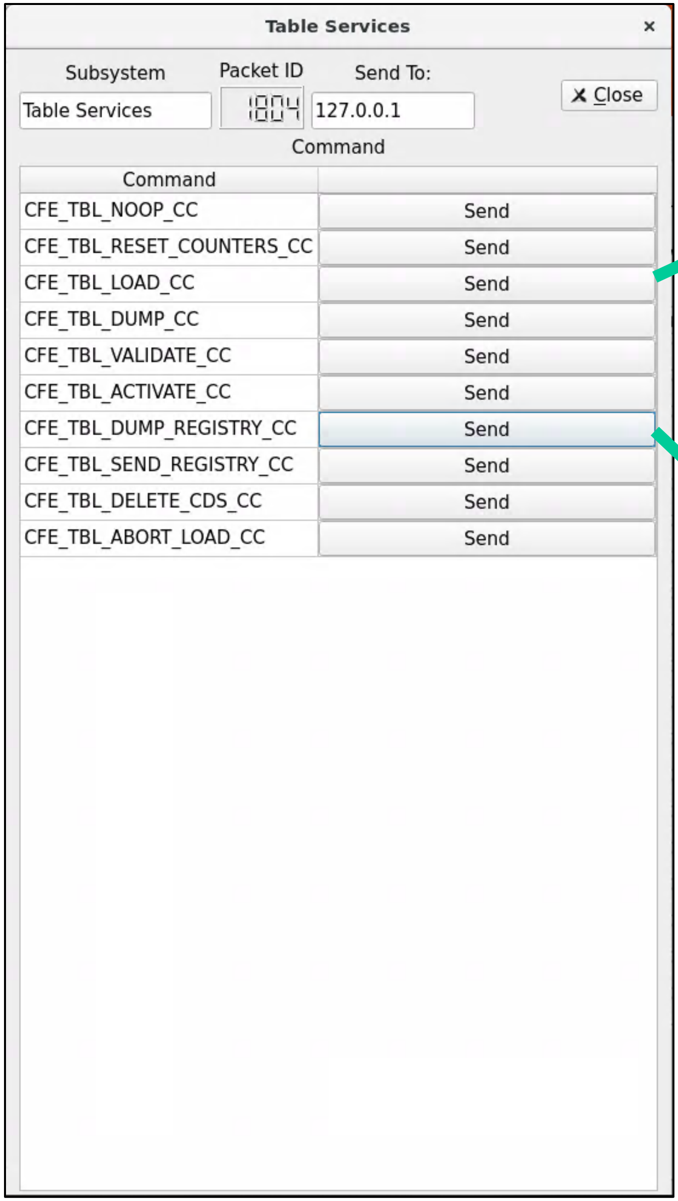


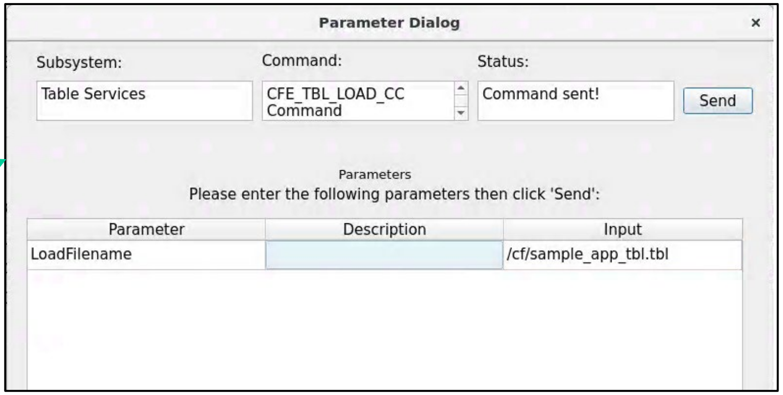
Table Services

Subsystem	Packet ID	Send To:
Table Services	1804	127.0.0.1

Command

Command	
CFE_TBL_NOOP_CC	Send
CFE_TBL_RESET_COUNTERS_CC	Send
CFE_TBL_LOAD_CC	Send
CFE_TBL_DUMP_CC	Send
CFE_TBL_VALIDATE_CC	Send
CFE_TBL_ACTIVATE_CC	Send
CFE_TBL_DUMP_REGISTRY_CC	Send
CFE_TBL_SEND_REGISTRY_CC	Send
CFE_TBL_DELETE_CDS_CC	Send
CFE_TBL_ABORT_LOAD_CC	Send

6



Parameter Dialog

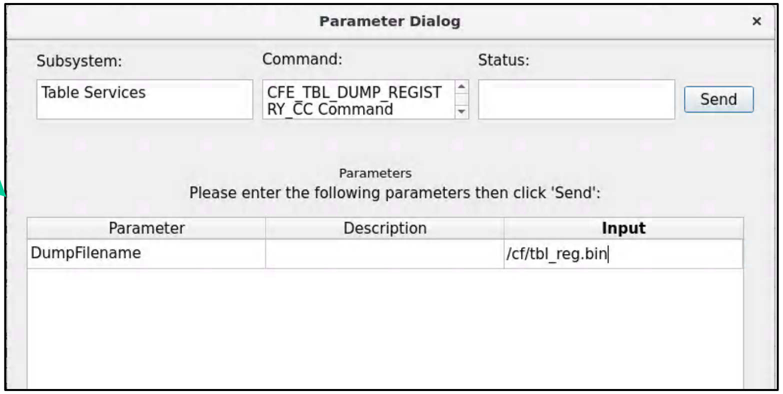
Subsystem:	Command:	Status:
Table Services	CFE_TBL_LOAD_CC Command	Command sent!

Parameters

Please enter the following parameters then click 'Send':

Parameter	Description	Input
LoadFilename		/cf/sample_app_tbl.tbl

7



Parameter Dialog

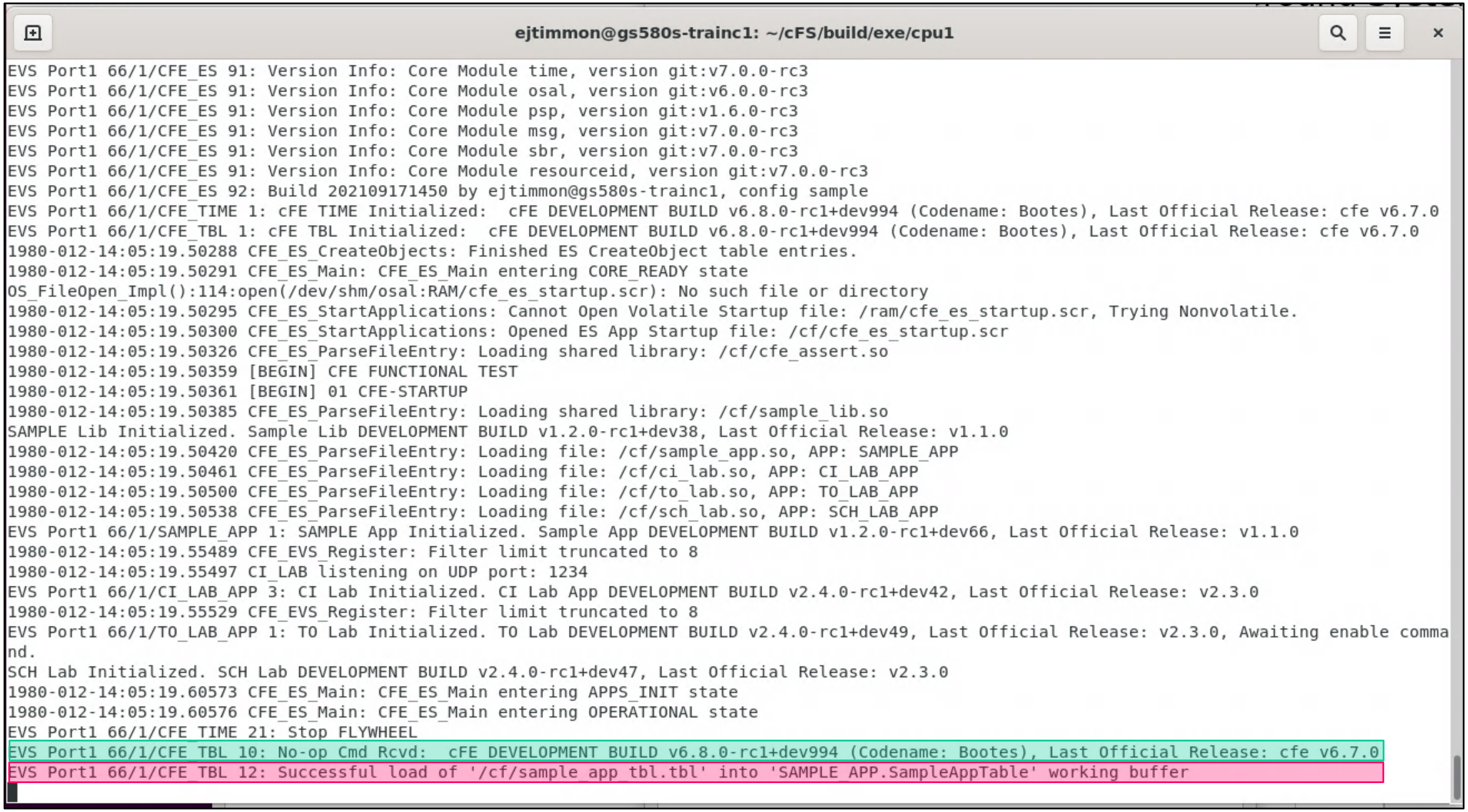
Subsystem:	Command:	Status:
Table Services	CFE_TBL_DUMP_REGISTRY_CC Command	

Parameters

Please enter the following parameters then click 'Send':

Parameter	Description	**Input**
DumpFilename		/cf/tbl_reg.bin

Exercise 6 - Recap

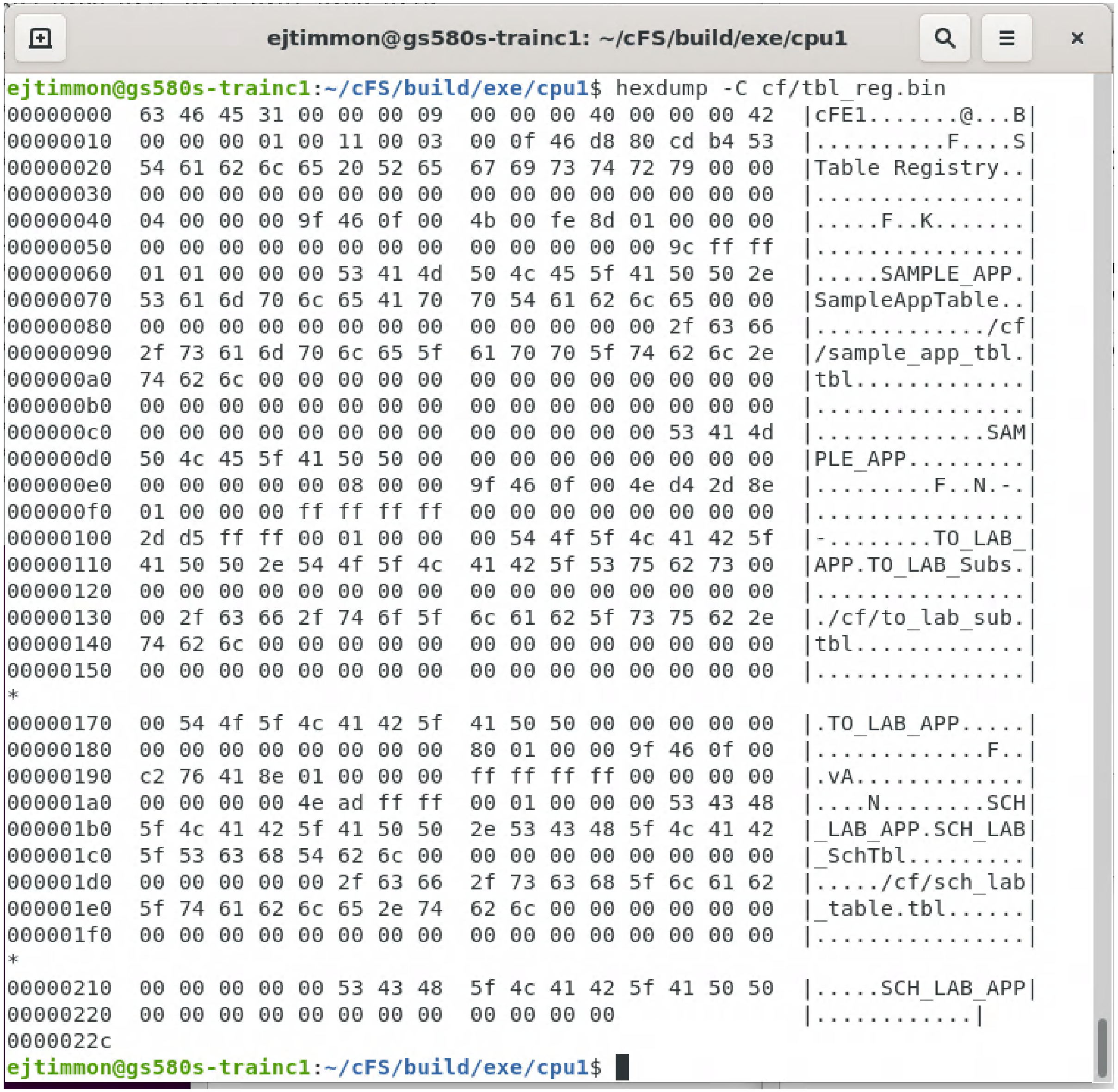


```
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module time, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module osal, version git:v6.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module psp, version git:v1.6.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module msg, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module sbr, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 91: Version Info: Core Module resourceid, version git:v7.0.0-rc3
EVS Port1 66/1/CFE_ES 92: Build 202109171450 by ejtimmon@gs580s-trainc1, config sample
EVS Port1 66/1/CFE_TIME 1: cFE TIME Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 1: cFE TBL Initialized:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
1980-012-14:05:19.50288 CFE_ES_CreateObjects: Finished ES CreateObject table entries.
1980-012-14:05:19.50291 CFE_ES_Main: CFE_ES_Main entering CORE_READY state
OS_FileOpen_Impl():114:open(/dev/shm/osal:RAM/cfe_es_startup.scr): No such file or directory
1980-012-14:05:19.50295 CFE_ES_StartApplications: Cannot Open Volatile Startup file: /ram/cfe_es_startup.scr, Trying Nonvolatile.
1980-012-14:05:19.50300 CFE_ES_StartApplications: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:05:19.50326 CFE_ES_ParseFileEntry: Loading shared library: /cf/cfe_assert.so
1980-012-14:05:19.50359 [BEGIN] CFE FUNCTIONAL TEST
1980-012-14:05:19.50361 [BEGIN] 01 CFE-STARTUP
1980-012-14:05:19.50385 CFE_ES_ParseFileEntry: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.2.0-rc1+dev38, Last Official Release: v1.1.0
1980-012-14:05:19.50420 CFE_ES_ParseFileEntry: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:05:19.50461 CFE_ES_ParseFileEntry: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
1980-012-14:05:19.50500 CFE_ES_ParseFileEntry: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:05:19.50538 CFE_ES_ParseFileEntry: Loading file: /cf/sch_lab.so, APP: SCH_LAB_APP
EVS Port1 66/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.2.0-rc1+dev66, Last Official Release: v1.1.0
1980-012-14:05:19.55489 CFE_EVS_Register: Filter limit truncated to 8
1980-012-14:05:19.55497 CI_LAB listening on UDP port: 1234
EVS Port1 66/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.4.0-rc1+dev42, Last Official Release: v2.3.0
1980-012-14:05:19.55529 CFE_EVS_Register: Filter limit truncated to 8
EVS Port1 66/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.4.0-rc1+dev49, Last Official Release: v2.3.0, Awaiting enable comma
nd.
SCH Lab Initialized. SCH Lab DEVELOPMENT BUILD v2.4.0-rc1+dev47, Last Official Release: v2.3.0
1980-012-14:05:19.60573 CFE_ES_Main: CFE_ES_Main entering APPS_INIT state
1980-012-14:05:19.60576 CFE_ES_Main: CFE_ES_Main entering OPERATIONAL state
EVS Port1 66/1/CFE_TIME 21: Stop FLYWHEEL
EVS Port1 66/1/CFE_TBL 10: No-op Cmd Rcvd:  cFE DEVELOPMENT BUILD v6.8.0-rc1+dev994 (Codename: Bootes), Last Official Release: cfe v6.7.0
EVS Port1 66/1/CFE_TBL 12: Successful load of '/cf/sample_app_tbl.tbl' into 'SAMPLE_APP.SampleAppTable' working buffer
```

TBL No-Op Command

Tbl Load Command

Exercise 6 - Recap



```
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1$ hexdump -C cf/tbl_reg.bin
00000000  63 46 45 31 00 00 00 09  00 00 00 40 00 00 00 42  |cFE1.......@...B|
00000010  00 00 00 01 00 11 00 03  00 0f 46 d8 80 cd b4 53  |..........F....S|
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 9f 46 0f 00  4b 00 fe 8d 01 00 00 00  |.....F..K.......|
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 41 70  70 54 61 62 6c 65 00 00  |SampleAppTable..|
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  61 70 70 5f 74 62 6c 2e  |/sample_app_tbl.|
000000a0  74 62 6c 00 00 00 00 00  00 00 00 00 00 00 00 00  |tbl.............|
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 00 00 00 08 00 00  9f 46 0f 00 4e d4 2d 8e  |.........F..N.-.|
000000f0  01 00 00 00 ff ff ff ff  00 00 00 00 00 00 00 00  |................|
00000100  2d d5 ff ff 00 01 00 00  00 54 4f 5f 4c 41 42 5f  |-........TO_LAB_|
00000110  41 50 50 2e 54 4f 5f 4c  41 42 5f 53 75 62 73 00  |APP.TO_LAB_Subs.|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000130  00 2f 63 66 2f 74 6f 5f  6c 61 62 5f 73 75 62 2e  |./cf/to_lab_sub.|
00000140  74 62 6c 00 00 00 00 00  00 00 00 00 00 00 00 00  |tbl.............|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000170  00 54 4f 5f 4c 41 42 5f  41 50 50 00 00 00 00 00  |.TO_LAB_APP.....|
00000180  00 00 00 00 00 00 00 00  80 01 00 00 9f 46 0f 00  |.............F..|
00000190  c2 76 41 8e 01 00 00 00  ff ff ff ff 00 00 00 00  |.vA.............|
000001a0  00 00 00 00 4e ad ff ff  00 01 00 00 00 53 43 48  |....N........SCH|
000001b0  5f 4c 41 42 5f 41 50 50  2e 53 43 48 5f 4c 41 42  |_LAB_APP.SCH_LAB|
000001c0  5f 53 63 68 54 62 6c 00  00 00 00 00 00 00 00 00  |_SchTbl.........|
000001d0  00 00 00 00 00 2f 63 66  2f 73 63 68 5f 6c 61 62  |...../cf/sch_lab|
000001e0  5f 74 61 62 6c 65 2e 74  62 6c 00 00 00 00 00 00  |_table.tbl......|
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000210  00 00 00 00 00 53 43 48  5f 4c 41 42 5f 41 50 50  |.....SCH_LAB_APP|
00000220  00 00 00 00 00 00 00 00  00 00 00 00              |............|
0000022c
ejtimmon@gs580s-trainc1:~/cFS/build/exe/cpu1$ 
```

3 Tables in System

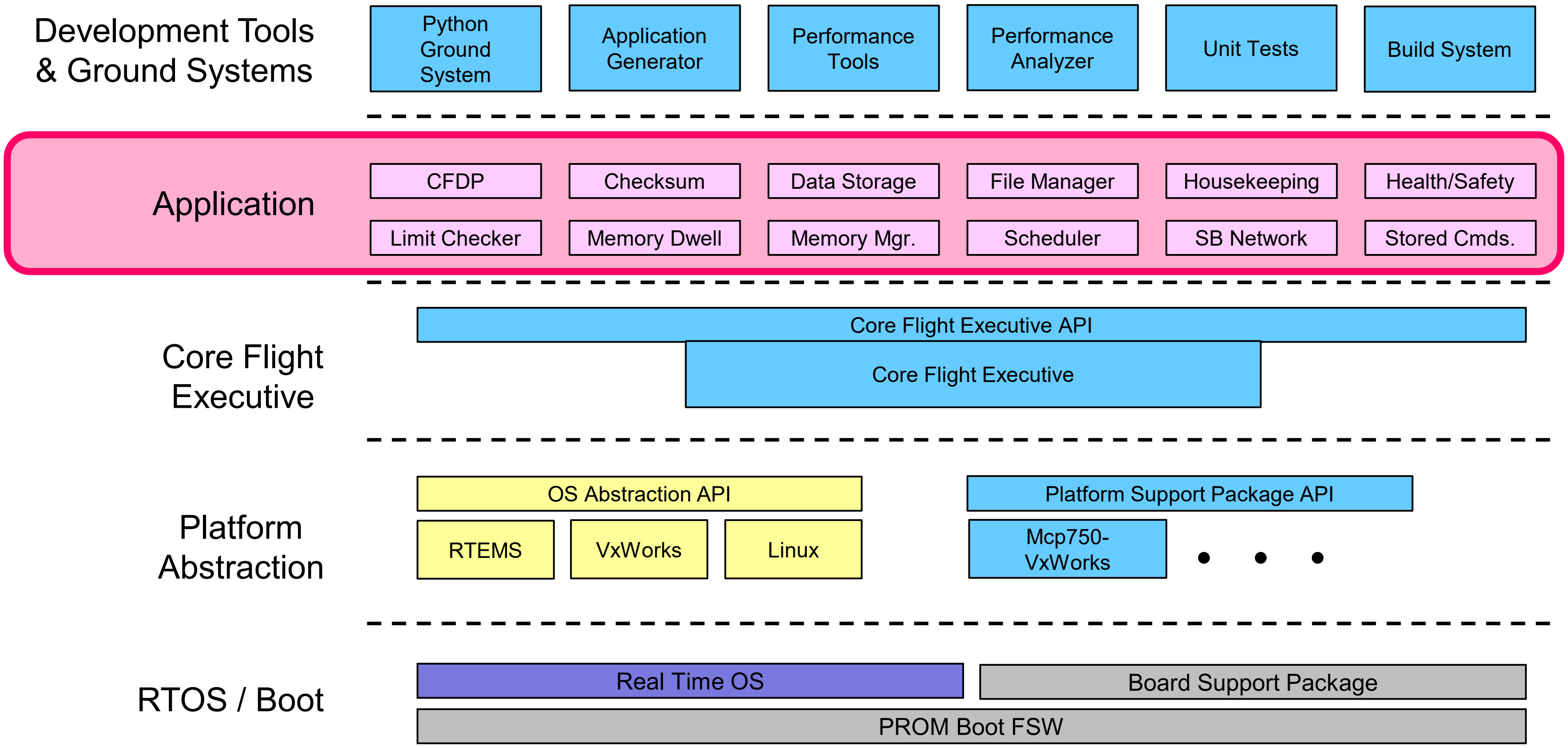
Core Flight System (cFS) Training

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

Applications - cFS Context



Development Tools & Ground Systems

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

Application

CFDP | Checksum | Data Storage | File Manager | Housekeeping | Health/Safety

Limit Checker | Memory Dwell | Memory Mgr. | Scheduler | SB Network | Stored Cmds.

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

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

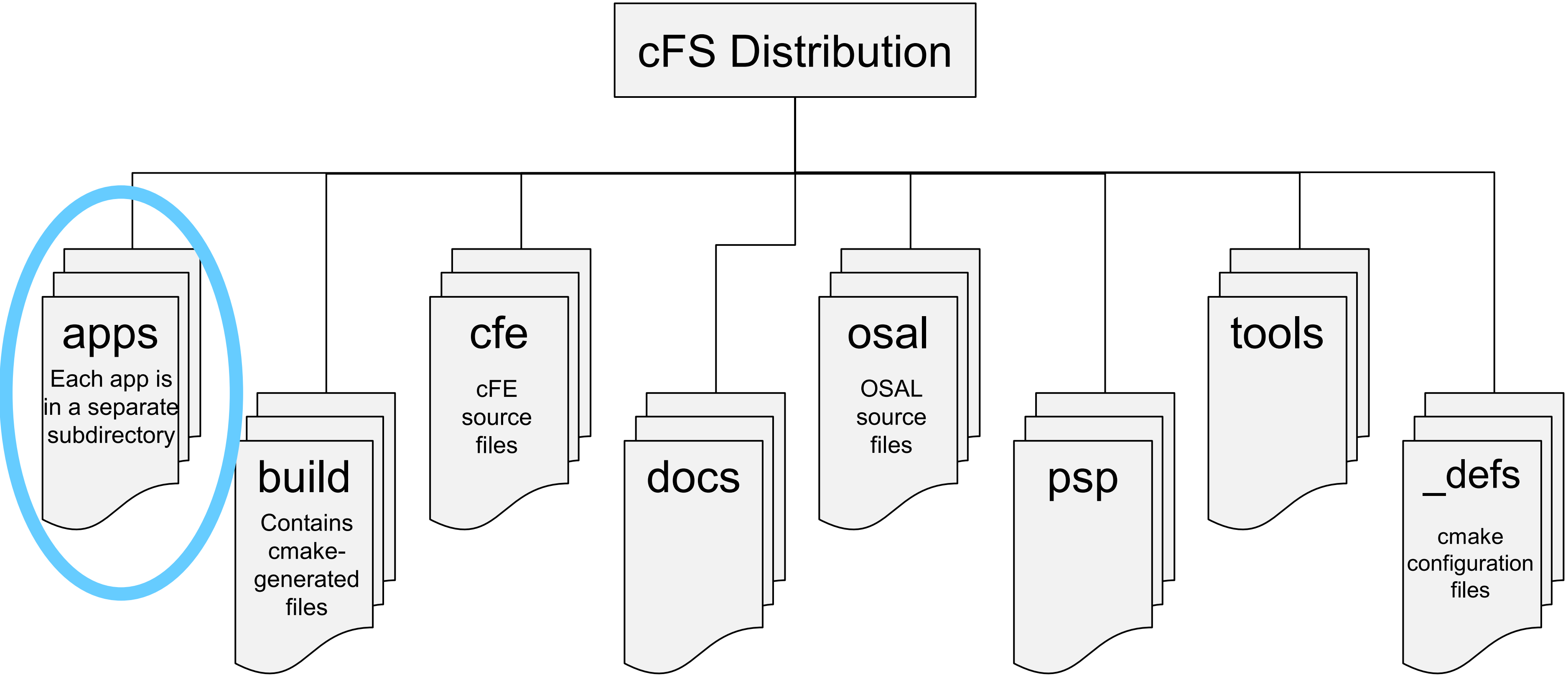
- **Missions use a combination of custom and reused applications**

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

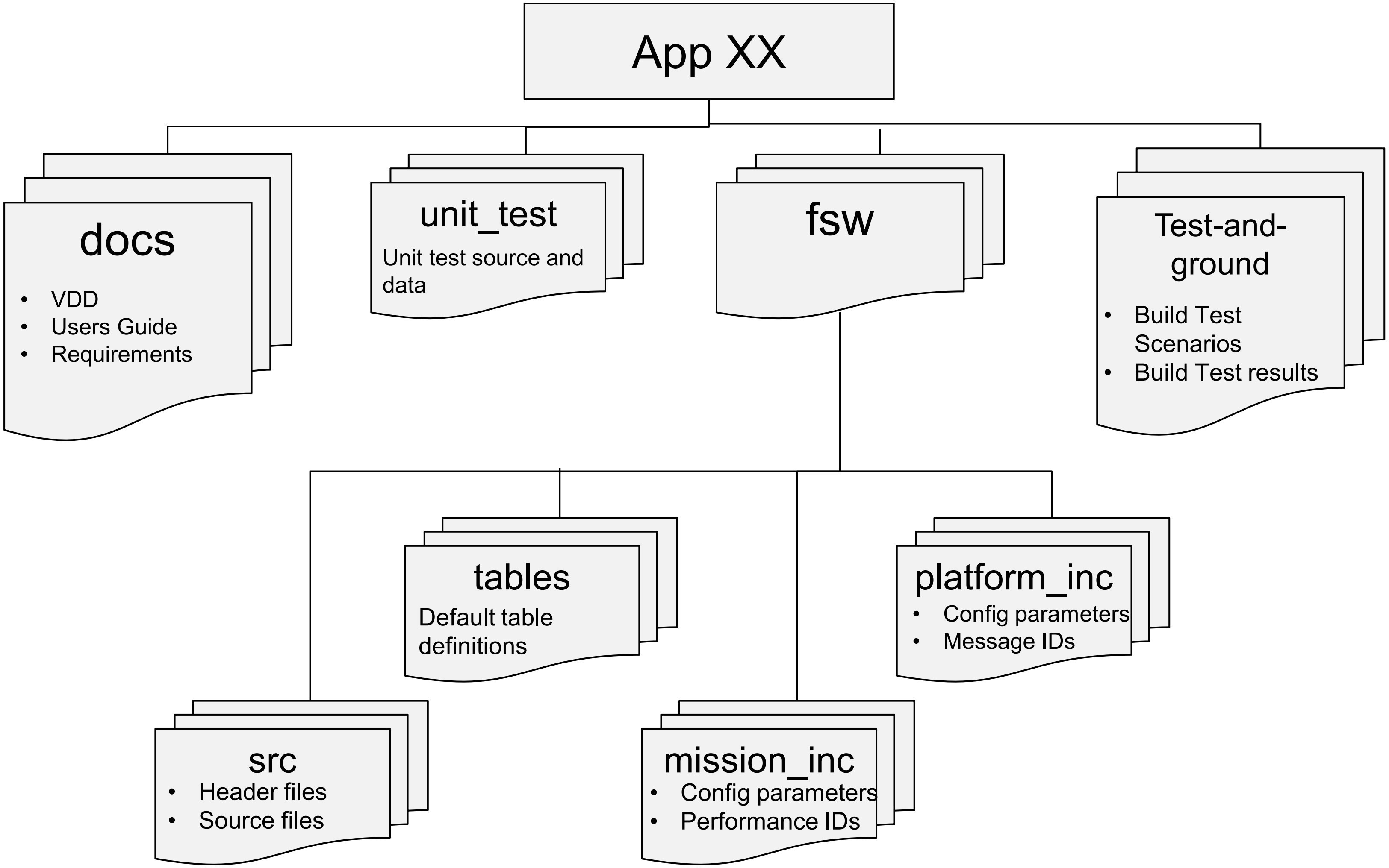
psp

tools

_defs

cmake configuration files

App Directory Structure



App XX

docs
- VDD
- Users Guide
- Requirements

unit_test

Unit test source and data

fsw

Test-and-ground
- Build Test Scenarios
- Build Test results

src
- Header files
- Source files

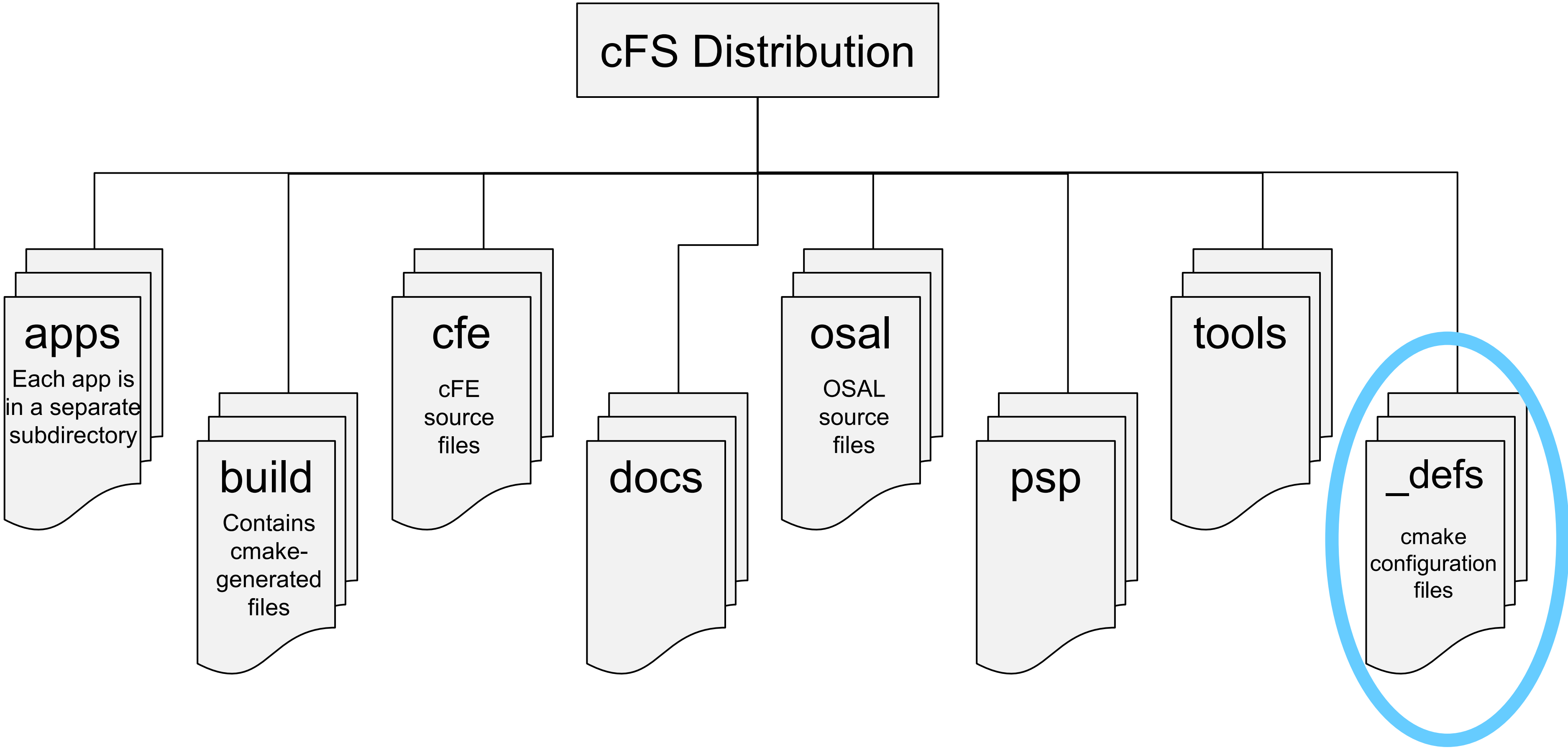
tables

Default table definitions

mission_inc
- Config parameters
- Performance IDs

platform_inc
- Config parameters
- Message IDs

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

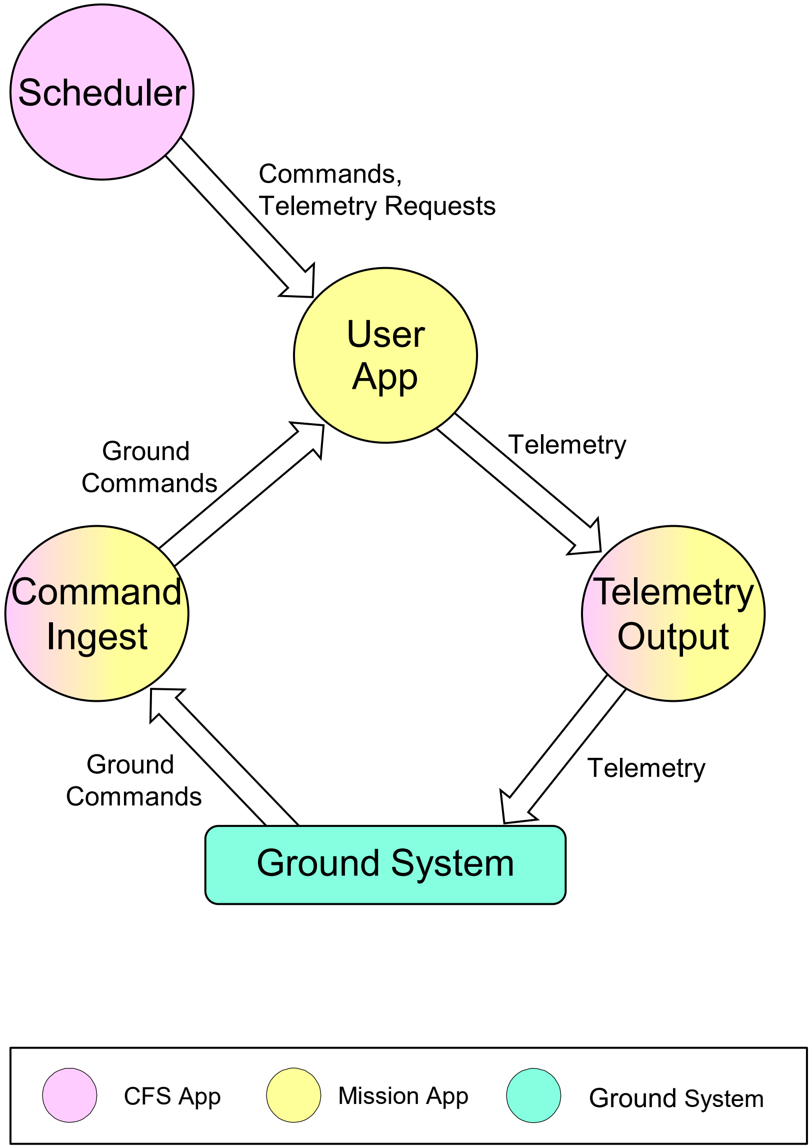
_def Directory Structure

- **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

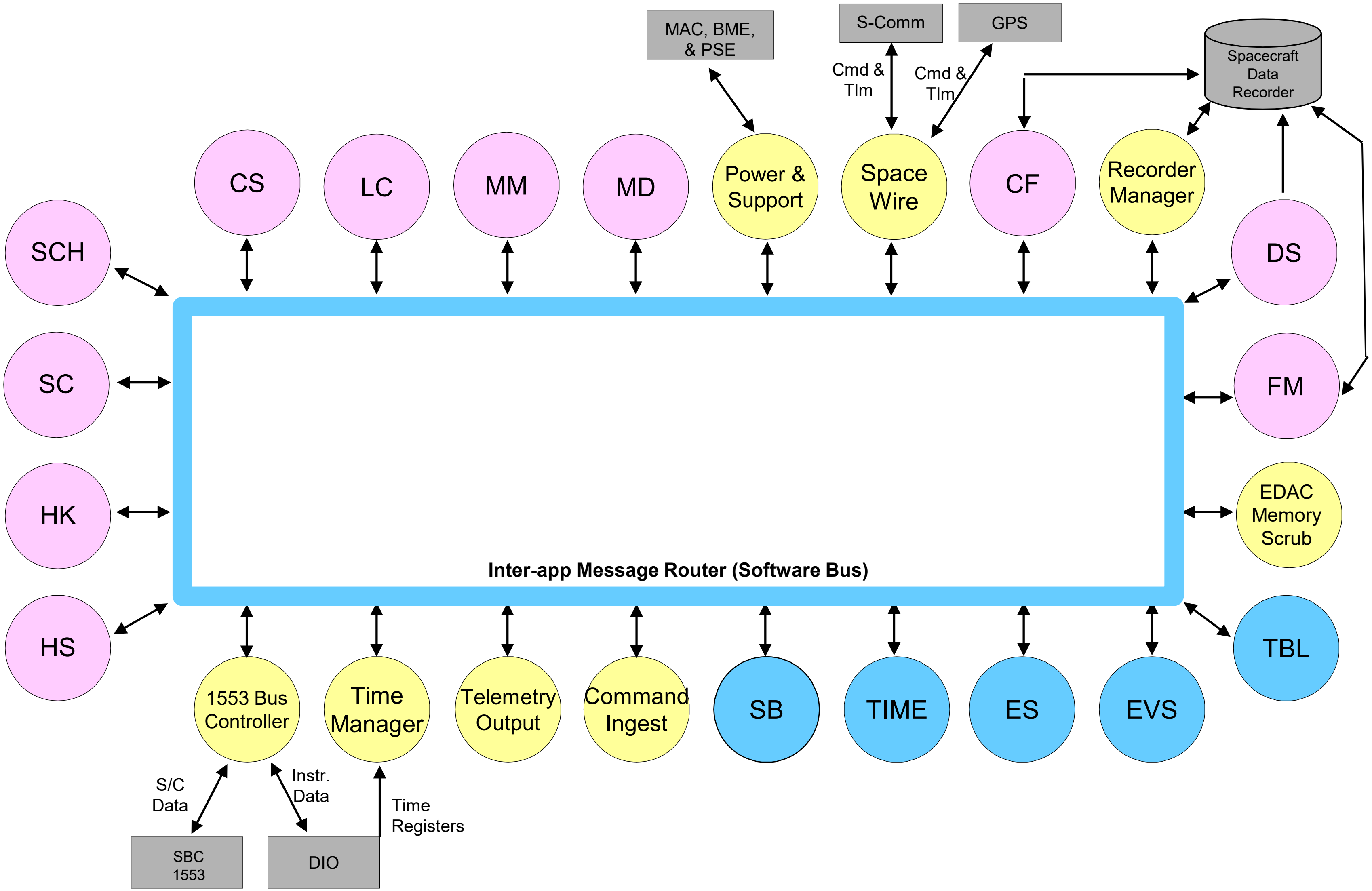
- **SCH, CI, and TO provide a runtime context that can be tailored for a particular environment**

- **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 the software bus and sends them to an external source, typically the ground system

Mission Application Example



● CFS App ● cFE Service ● Mission C&DH App ● Hardware

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**

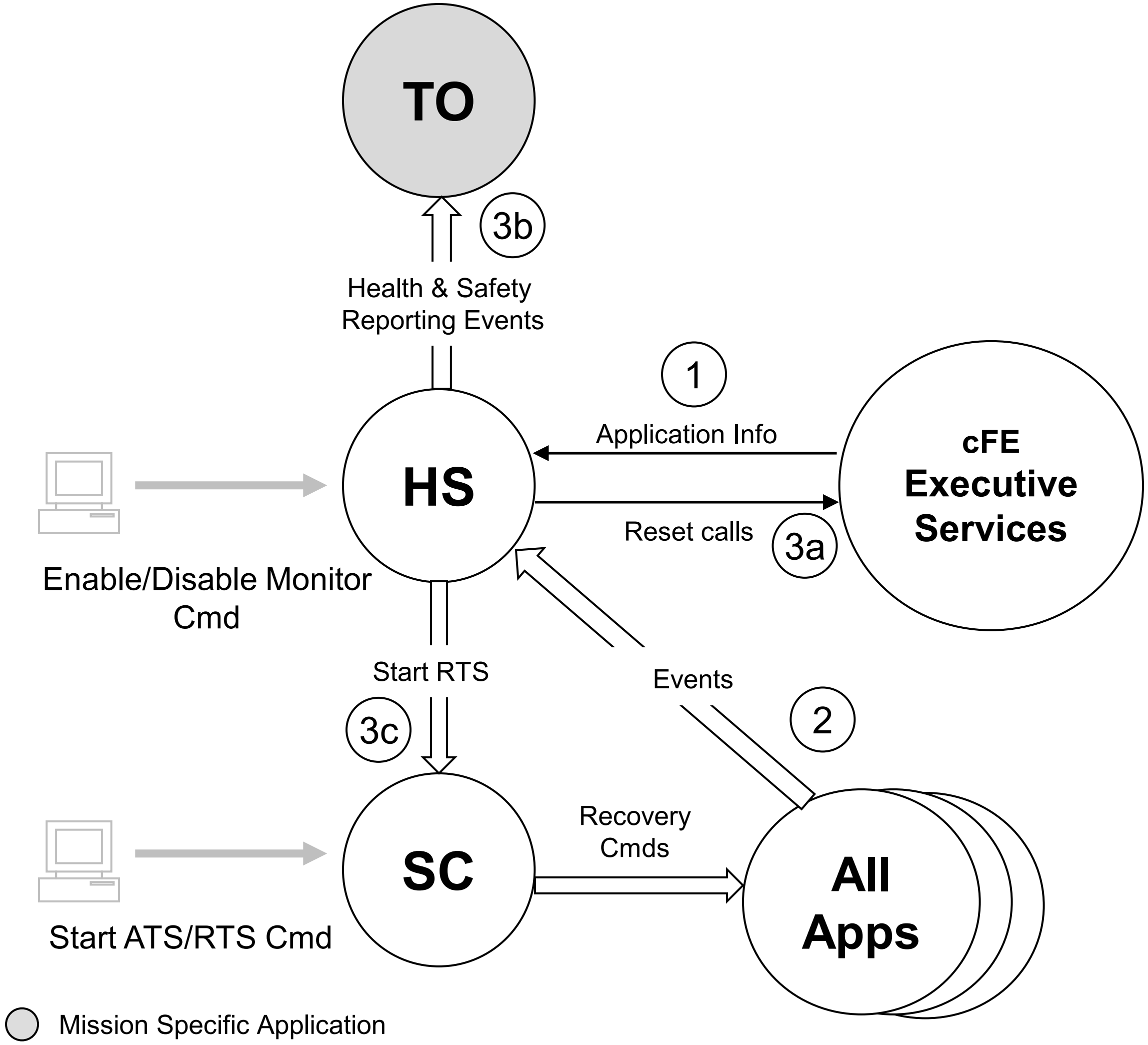
- **Health & Safety (HS) – Ensures critical tasks check-in, services watchdog, detects CPU hogging, calculates CPU utilization**

- **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

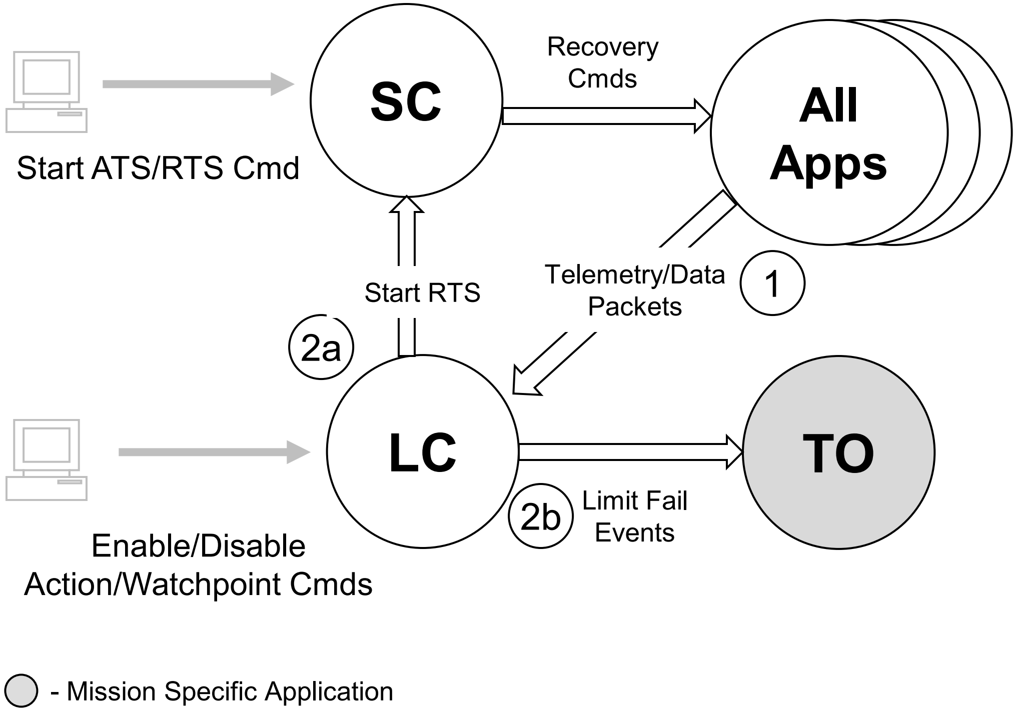
1) **HS monitors applications**
2) **HS monitors event messages**
3) **HS Table specified actions are taken in response to application and event monitoring:**
   a) **Reset applications or the processor**
   b) **Send Event message**
   c) **Initiate Stored Command (SC) recovery sequence**



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

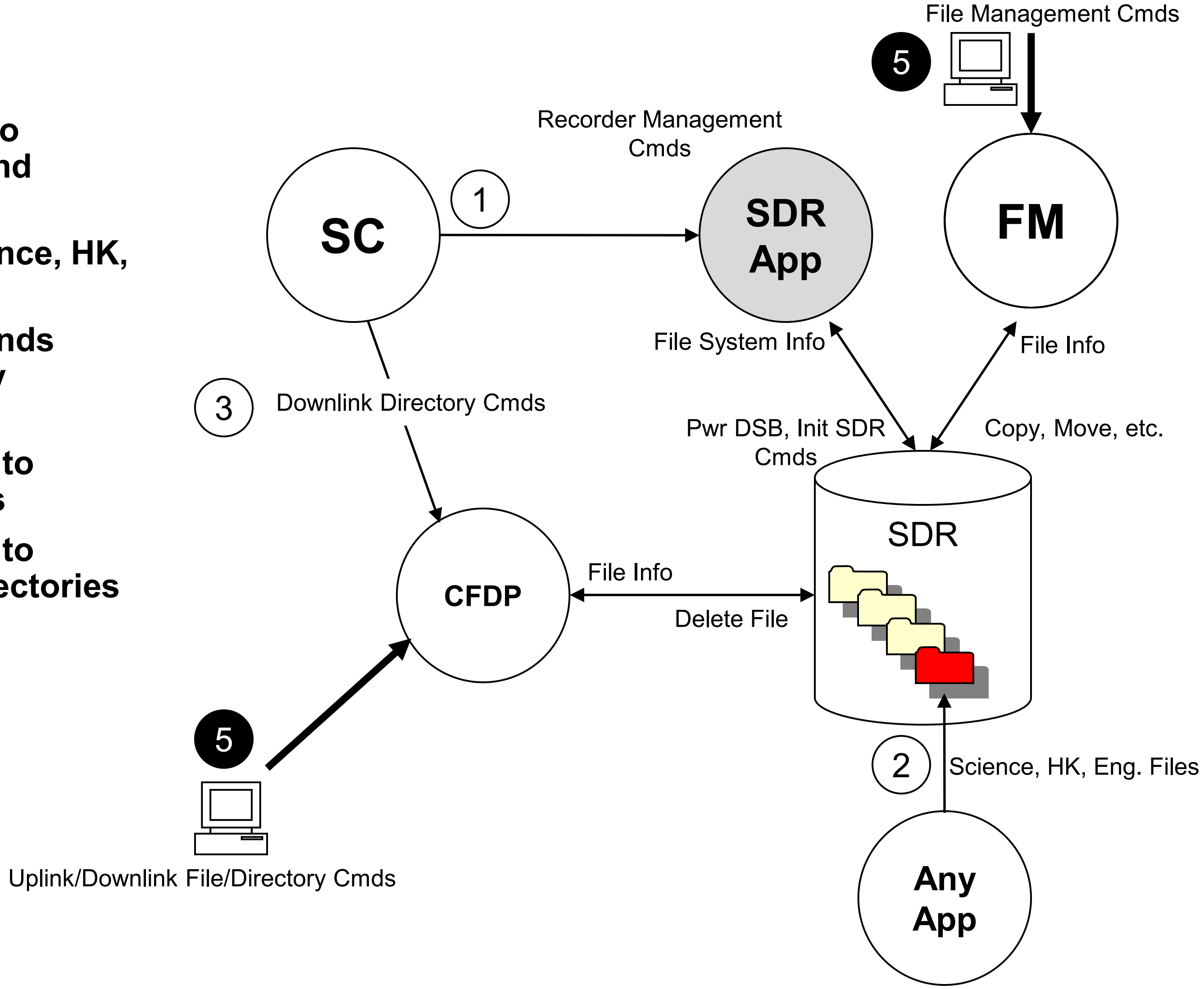


File & Data Management Apps

- 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

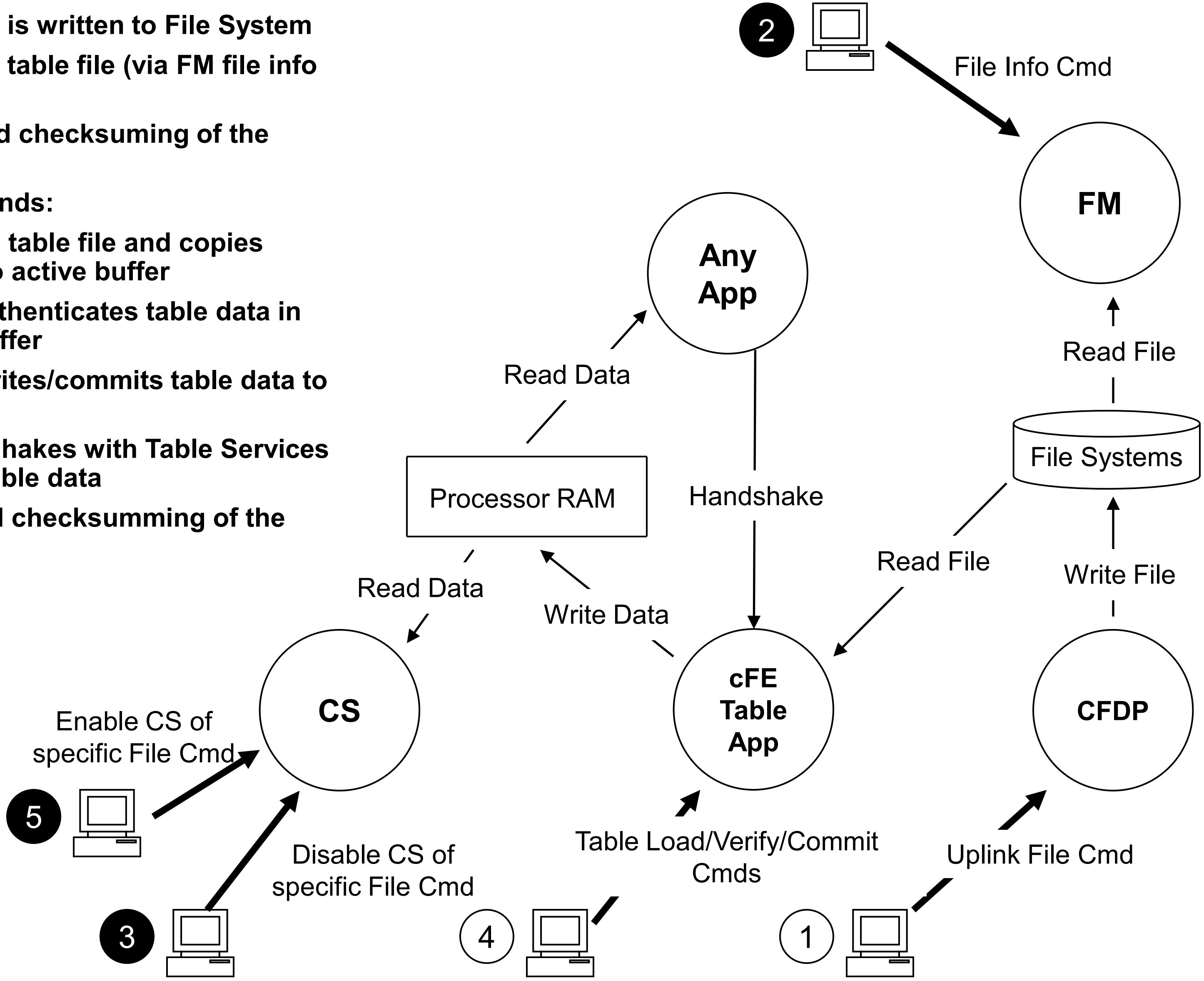
○ - Mission Specific Application

● - 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

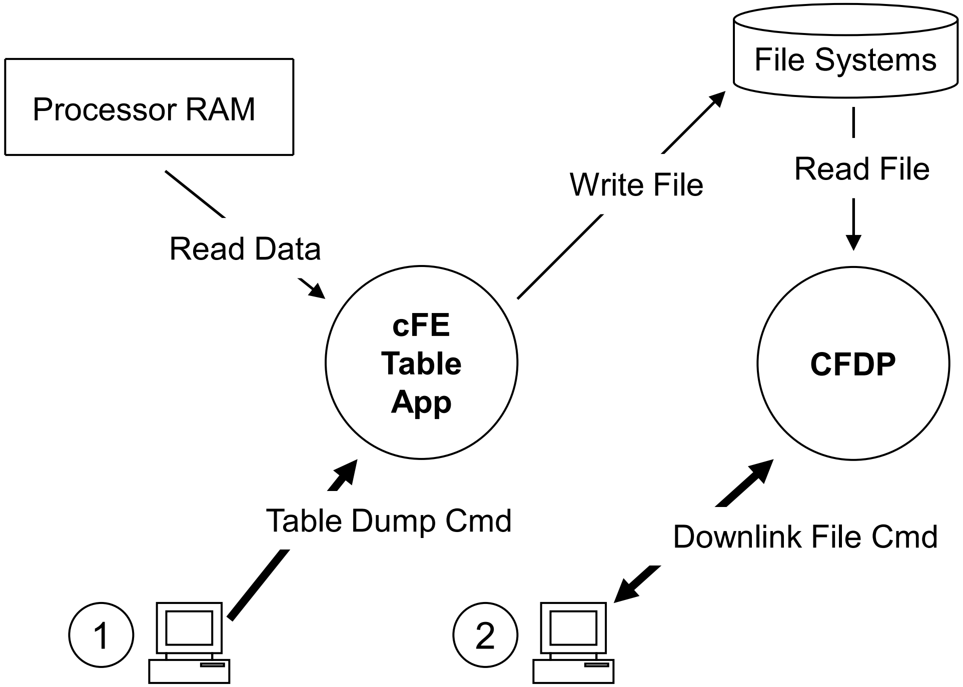
   Application handshakes with Table Services to read updated table data
5) Enable background checksumming of the table



● - 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**

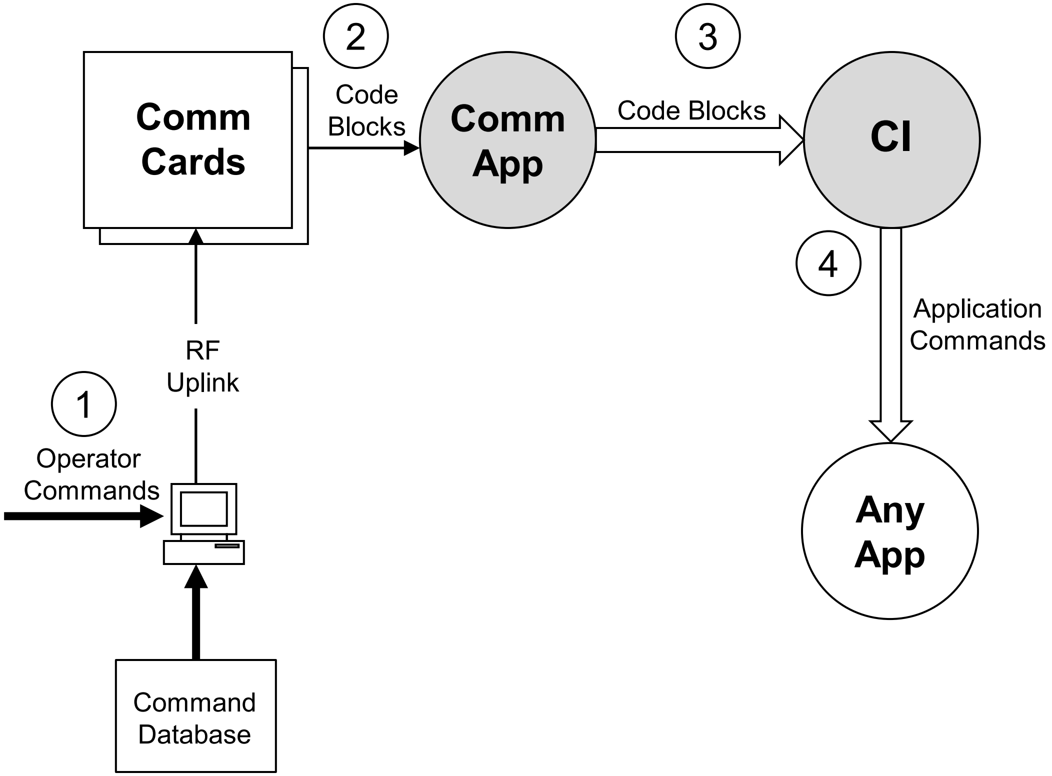
- **Command Ingest (CI) – Receives ground commands, validates them, and distributes them throughout the system; this app is often custom**

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

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

Operational Scenarios
Uplink

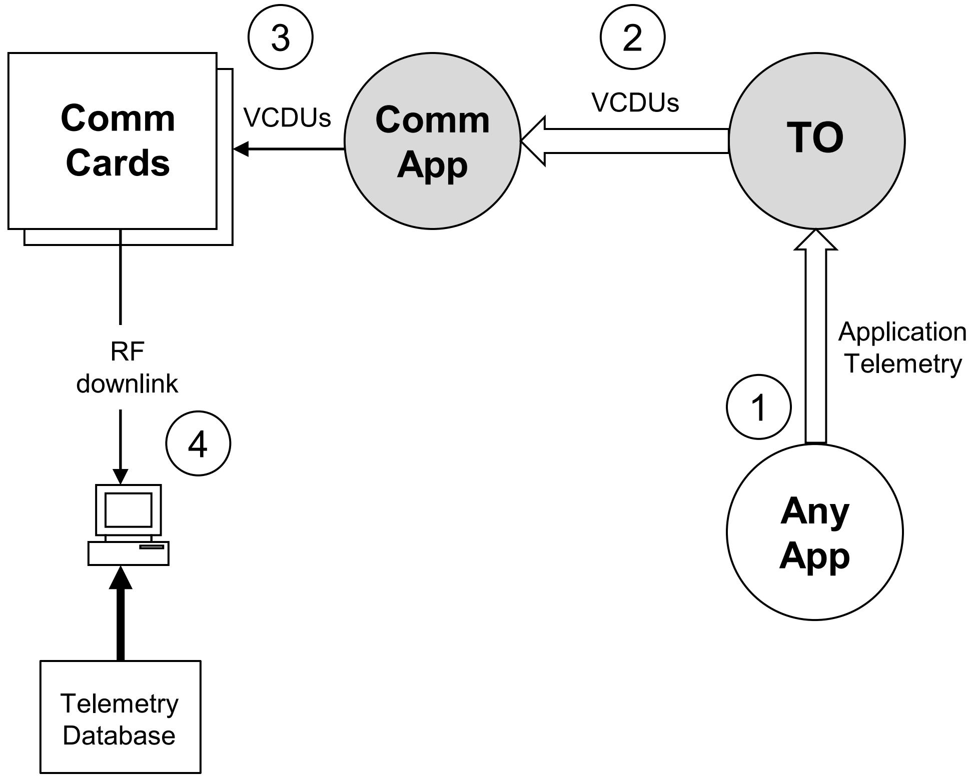
1) **Commands sent from ground system are received by communication hardware**
2) **Communication hardware processes commands received and sends code blocks to receiving application.**
3) **Communication application strips off any hardware protocol wrappers, packages Code Blocks for transfer over software 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

Application Design

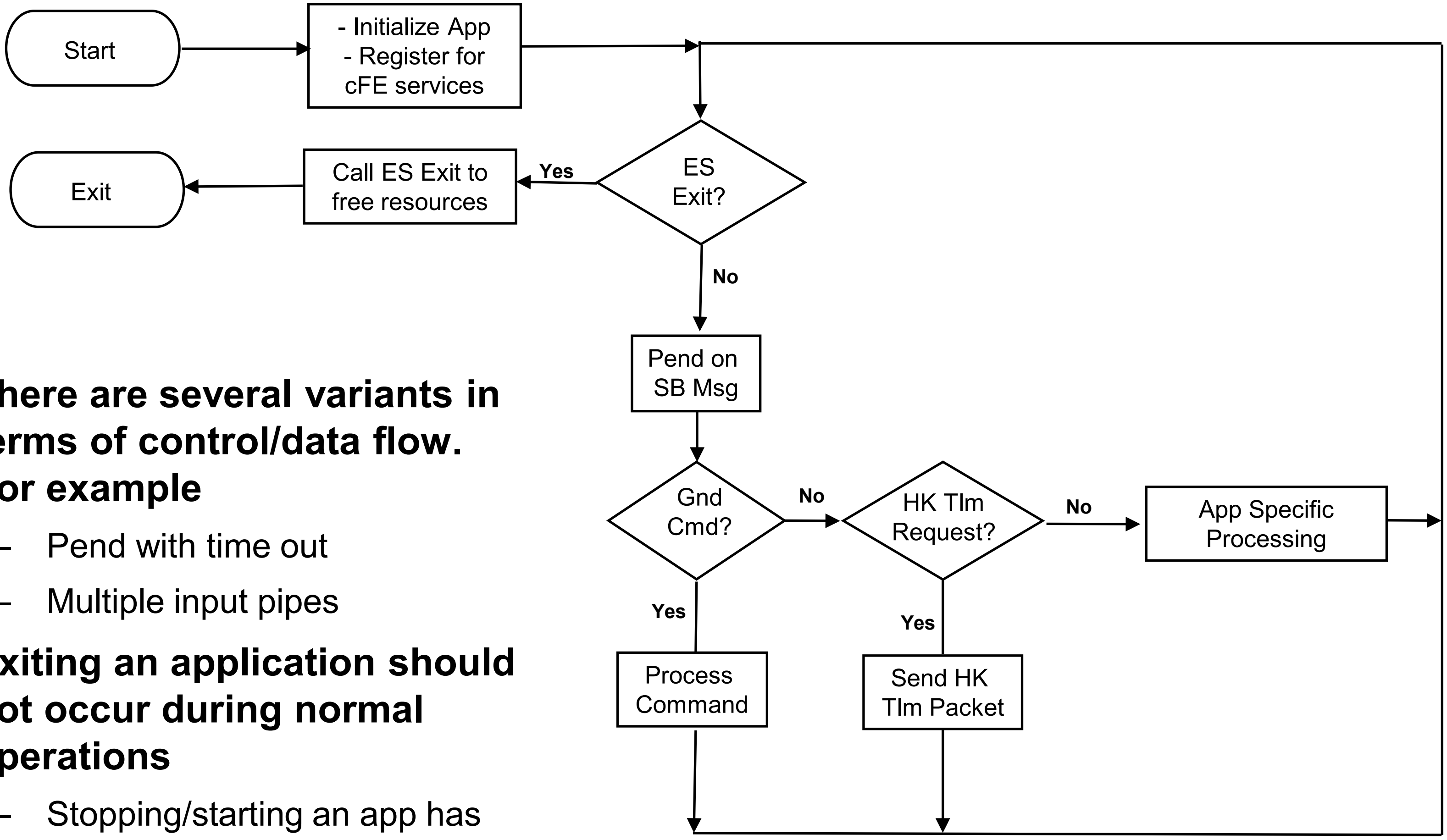
Application Design Resources

- **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

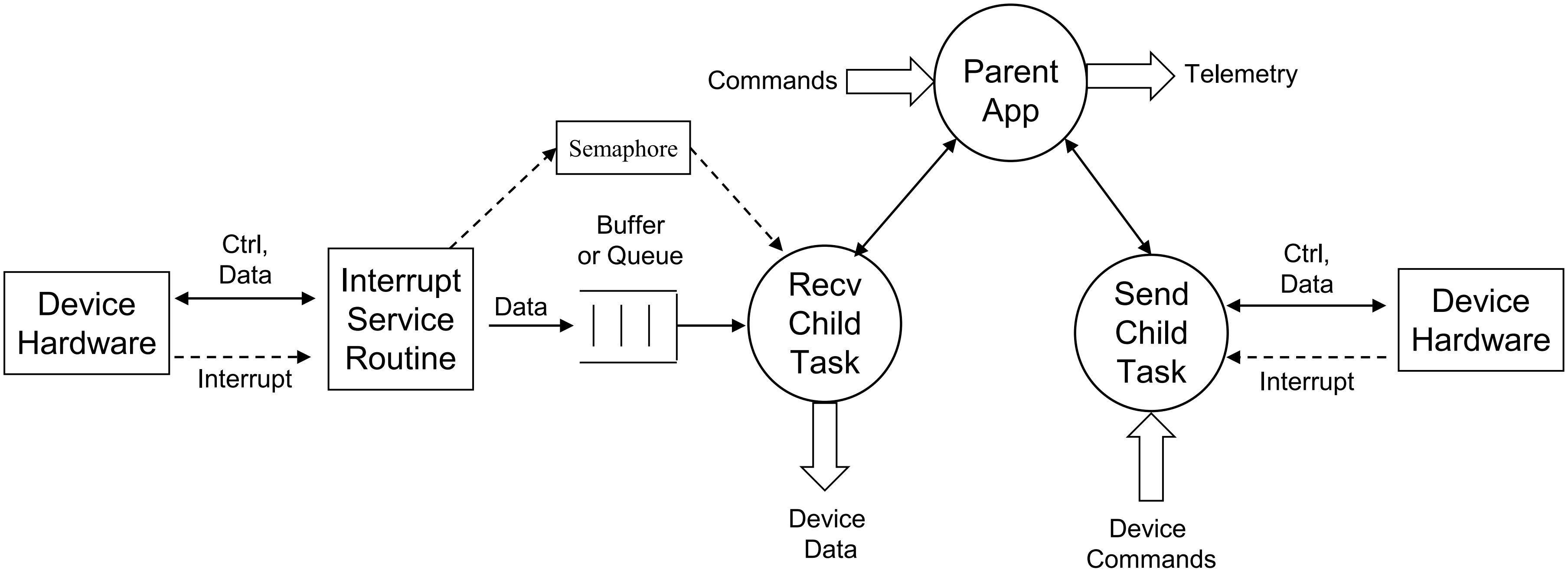
- **Allocate resources during initialization to help keep run loop deterministic**
- **Use a lower priority child task for long operations like a memory dump**
  - Create child tasks during initialization
- **Register with EVS immediately after registering app so local event log can be used instead of 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 7 - Add a command to sample_app

Part 1 – Add new command code event message

1. Navigate to 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_APP_HELLO_WORLD_INF_EID           8
#define SAMPLE_APP_EVENT_COUNTS                  8
```

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

```
SAMPLE_APP_Data.EventFilters[7].EventID = SAMPLE_APP_HELLO_WORLD_INF_EID;
SAMPLE_APP_Data.EventFilters[7].Mask    = 0x0000;
```

5. In sample_app.c, add a case for the new command code in SAMPLE_APP_ProcessGroundCommand

```
case SAMPLE_APP_HELLO_WORLD_CC:
    if (SAMPLE_APP_VerifyCmdLength(&SBBufPtr->Msg, sizeof(SAMPLE_APP_NoopCmd_t))) {
        SAMPLE_APP_HelloCmd((SAMPLE_APP_NoopCmd_t * )SBBufPtr);
    }
    break;
```

Exercise 7 - Add a command to sample_app

Part 2 – Add code to handle new command

6. In sample_app.c, add a new function called SAMPLE_HelloCmd

```
int32 SAMPLE_APP_HelloCmd( const SAMPLE_APP_NoopCmd_t * Msg )  {
    SAMPLE_APP_Data.CmdCounter++;
    CFE_EVS_SendEvent(SAMPLE_APP_HELLO_WORLD_INF_EID,
                      CFE_EVS_EventType_INFORMATION,
                      "Hello, World.  This is sample_app!");
    return CFE_SUCCESS;
}
```

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

```
int32  SAMPLE_APP_HelloCmd(const SAMPLE_APP_NoopCmd_t * Msg);
```

8. Recompile cFS

```
make
make install
```

Exercise 7 - Add a command to sample_app

Part 3 – Add ground command to GroundSystem.py

1. Navigate to the /cmdGui directory from the top level cFS directory

```
cd tools/cFS-GroundSystem/Subsystems/cmdGui
```

2. Open the CHeaderParser-hdr-paths.txt and uncomment only the ‘`sample_app_msg.h`’ line

```
#../../../../apps/to_lab/fsw/src/to_lab_msg.h
#../../../../apps/ci_lab/fsw/src/ci_lab_msg.h
../../../../apps/sample_app/fsw/src/sample_app_msg.h
#../../../../cfe/fsw/cfe-core/src/inc/cfe_es_msg.h
#../../../../cfe/fsw/cfe-core/src/inc/cfe_time_msg.h
```

3. Run the CHeaderParser.py script

```
python3 CHeaderParser.py
```

- When prompted, select a name for the command file to be saved as:

  Example: APPS_SAMPLE_APP_CMD

- Respond ‘no’ when asked if any of the commands require parameters.

Exercise 7 - Add a command to sample_app

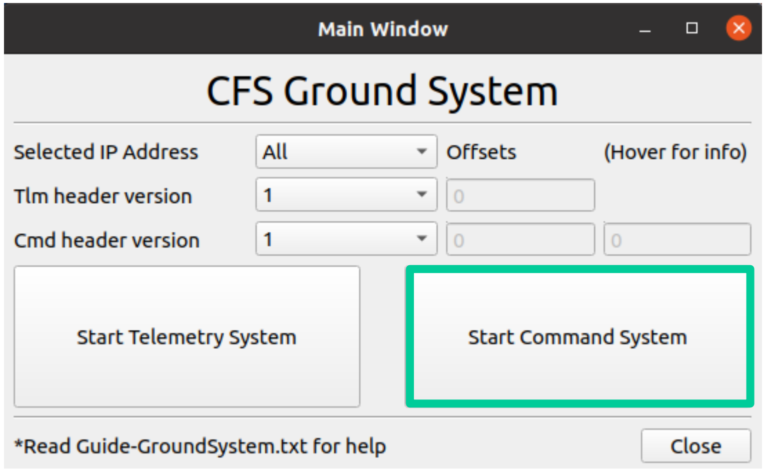
Part 3 – Add ground command to GroundSystem.py (continued)

4. Edit the command-pages.txt file to update the name of the SAMPLE_APP cmd file with the name chosen on step 3.

```
Command Ingest,   CI_LAB_CMD,          0x1884, LE, UdpCommands.py,  127.0.0.1,   1234
Telemetry Output, TO_LAB_CMD,          0x1880, LE, UdpCommands.py,  127.0.0.1,   1234
Sample App,       APPS_SAMPLE_APP_CMD, 0x1882, LE, UdpCommands.py,  127.0.0.1,   1234
Spare,                          ,      0x0000, LE, UdpCommands.py,  127.0.0.1,   1234
Spare,                          ,      0x0000, LE, UdpCommands.py,  127.0.0.1,   1234
```

5. Navigate to /cFS-GroundSystem and launch GroundSystem.py

```
        cd ../..
        python3 GroundSystem.py
```



Main Window

CFS Ground System

Selected IP Address: All

Offsets (Hover for info)

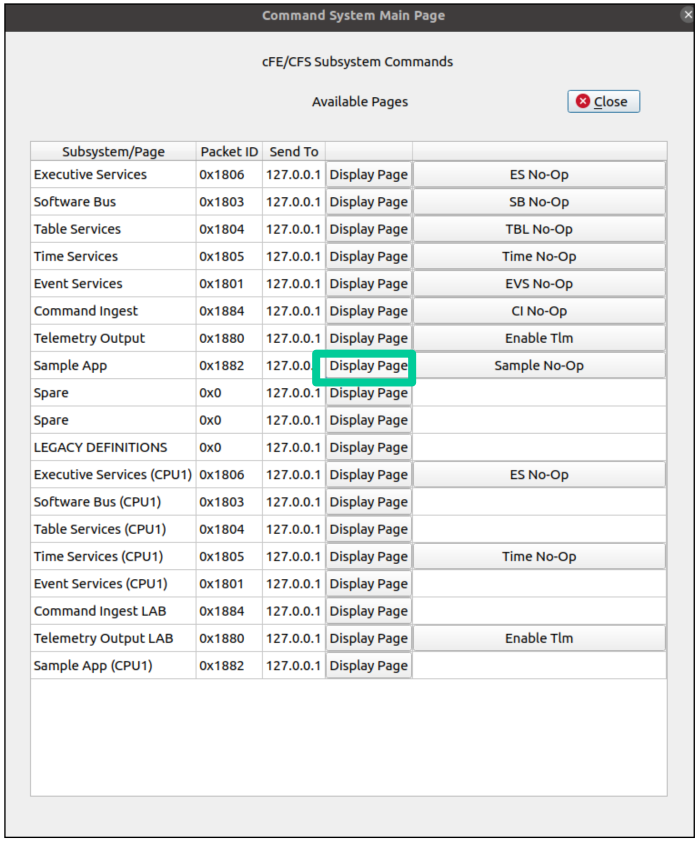
Tlm header version: 1 | 0

Cmd header version: 1 | 0 | 0

Exercise 7 - Add a command to sample_app

Part 3 – Add ground command to GroundSystem.py (continued)

6. Launch Sample App Command Display Page and Send Command

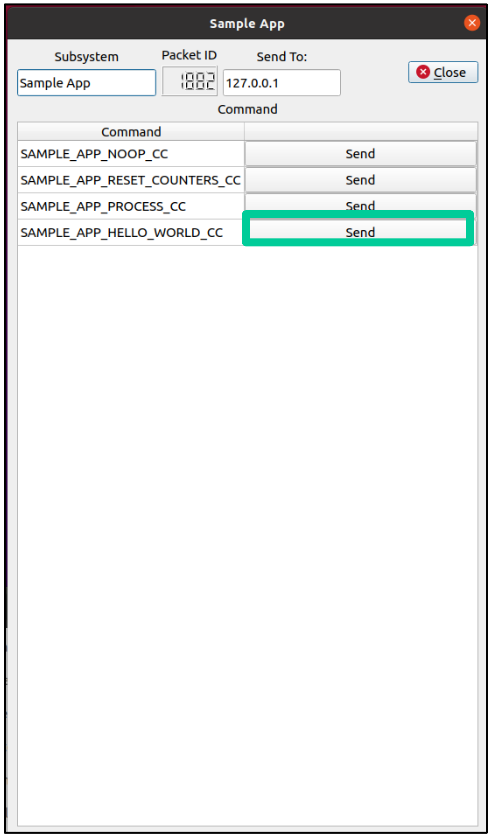


Command System Main Page

cFE/CFS Subsystem Commands

Available Pages

Subsystem/Page	Packet ID	Send To		
Executive Services	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus	0x1803	127.0.0.1	Display Page	SB No-Op
Table Services	0x1804	127.0.0.1	Display Page	TBL No-Op
Time Services	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services	0x1801	127.0.0.1	Display Page	EVS No-Op
Command Ingest	0x1884	127.0.0.1	Display Page	CI No-Op
Telemetry Output	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App	0x1882	127.0.0	Display Page	Sample No-Op
Spare	0x0	127.0.0.1	Display Page	
Spare	0x0	127.0.0.1	Display Page	
LEGACY DEFINITIONS	0x0	127.0.0.1	Display Page	
Executive Services (CPU1)	0x1806	127.0.0.1	Display Page	ES No-Op
Software Bus (CPU1)	0x1803	127.0.0.1	Display Page	
Table Services (CPU1)	0x1804	127.0.0.1	Display Page	
Time Services (CPU1)	0x1805	127.0.0.1	Display Page	Time No-Op
Event Services (CPU1)	0x1801	127.0.0.1	Display Page	
Command Ingest LAB	0x1884	127.0.0.1	Display Page	
Telemetry Output LAB	0x1880	127.0.0.1	Display Page	Enable Tlm
Sample App (CPU1)	0x1882	127.0.0.1	Display Page	



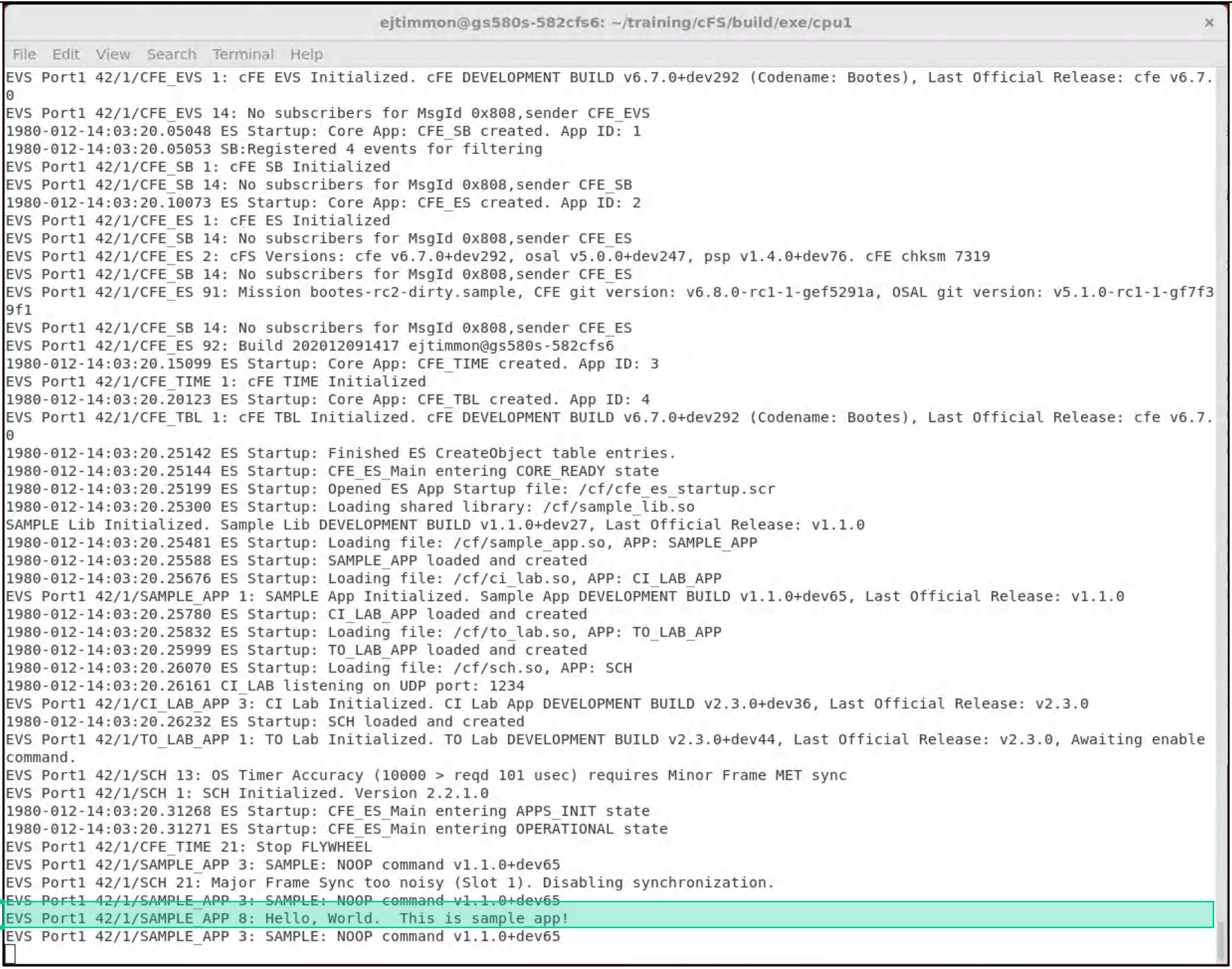
Sample App

Subsystem	Packet ID	Send To:
Sample App	1882	127.0.0.1

Command

Command	
SAMPLE_APP_NOOP_CC	Send
SAMPLE_APP_RESET_COUNTERS_CC	Send
SAMPLE_APP_PROCESS_CC	Send
SAMPLE_APP_HELLO_WORLD_CC	Send

Exercise 7 Recap



```
EVS Port1 42/1/CFE_EVS 1: cFE EVS Initialized. cFE DEVELOPMENT BUILD v6.7.0+dev292 (Codename: Bootes), Last Official Release: cfe v6.7.
0
EVS Port1 42/1/CFE_EVS 14: No subscribers for MsgId 0x808,sender CFE_EVS
1980-012-14:03:20.05048 ES Startup: Core App: CFE_SB created. App ID: 1
1980-012-14:03:20.05053 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
1980-012-14:03:20.10073 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: cFS Versions: cfe v6.7.0+dev292, osal v5.0.0+dev247, psp v1.4.0+dev76. cFE chksm 7319
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 91: Mission bootes-rc2-dirty.sample, CFE git version: v6.8.0-rc1-1-gef5291a, OSAL git version: v5.1.0-rc1-1-gf7f3
9f1
EVS Port1 42/1/CFE_SB 14: No subscribers for MsgId 0x808,sender CFE_ES
EVS Port1 42/1/CFE_ES 92: Build 202012091417 ejtimmon@gs580s-582cfs6
1980-012-14:03:20.15099 ES Startup: Core App: CFE_TIME created. App ID: 3
EVS Port1 42/1/CFE_TIME 1: cFE TIME Initialized
1980-012-14:03:20.20123 ES Startup: Core App: CFE_TBL created. App ID: 4
EVS Port1 42/1/CFE_TBL 1: cFE TBL Initialized. cFE DEVELOPMENT BUILD v6.7.0+dev292 (Codename: Bootes), Last Official Release: cfe v6.7.
0
1980-012-14:03:20.25142 ES Startup: Finished ES CreateObject table entries.
1980-012-14:03:20.25144 ES Startup: CFE_ES_Main entering CORE_READY state
1980-012-14:03:20.25199 ES Startup: Opened ES App Startup file: /cf/cfe_es_startup.scr
1980-012-14:03:20.25300 ES Startup: Loading shared library: /cf/sample_lib.so
SAMPLE Lib Initialized. Sample Lib DEVELOPMENT BUILD v1.1.0+dev27, Last Official Release: v1.1.0
1980-012-14:03:20.25481 ES Startup: Loading file: /cf/sample_app.so, APP: SAMPLE_APP
1980-012-14:03:20.25588 ES Startup: SAMPLE_APP loaded and created
1980-012-14:03:20.25676 ES Startup: Loading file: /cf/ci_lab.so, APP: CI_LAB_APP
EVS Port1 42/1/SAMPLE_APP 1: SAMPLE App Initialized. Sample App DEVELOPMENT BUILD v1.1.0+dev65, Last Official Release: v1.1.0
1980-012-14:03:20.25780 ES Startup: CI_LAB_APP loaded and created
1980-012-14:03:20.25832 ES Startup: Loading file: /cf/to_lab.so, APP: TO_LAB_APP
1980-012-14:03:20.25999 ES Startup: TO_LAB_APP loaded and created
1980-012-14:03:20.26070 ES Startup: Loading file: /cf/sch.so, APP: SCH
1980-012-14:03:20.26161 CI_LAB listening on UDP port: 1234
EVS Port1 42/1/CI_LAB_APP 3: CI Lab Initialized. CI Lab App DEVELOPMENT BUILD v2.3.0+dev36, Last Official Release: v2.3.0
1980-012-14:03:20.26232 ES Startup: SCH loaded and created
EVS Port1 42/1/TO_LAB_APP 1: TO Lab Initialized. TO Lab DEVELOPMENT BUILD v2.3.0+dev44, Last Official Release: v2.3.0, 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
1980-012-14:03:20.31268 ES Startup: CFE_ES_Main entering APPS_INIT state
1980-012-14:03:20.31271 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 v1.1.0+dev65
EVS Port1 42/1/SCH 21: Major Frame Sync too noisy (Slot 1). Disabling synchronization.
EVS Port1 42/1/SAMPLE_APP 3: SAMPLE: NOOP command v1.1.0+dev65
EVS Port1 42/1/SAMPLE_APP 8: Hello, World.  This is sample app!
EVS Port1 42/1/SAMPLE_APP 3: SAMPLE: NOOP command v1.1.0+dev65
```

Sample App 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
CCB	Configuration Control Board	EEPROM	Electrically Erasable Programmable Read-Only Memory
CCSDS	Consultative Committee for Space Data Systems	ES	Executive Services
CDS	Critical Data Store	EVS	Event Services
CESE	Center for Experimental Software Engineering	FDC	Failure Detection and Correction
CFDP	CCSDS File Delivery Protocol	FDIR	Failure Detection, Isolation, and Recovery
cFE	Core Flight Executive	FM	File Management, Fault Management
cFS	Core Flight Software System		

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