

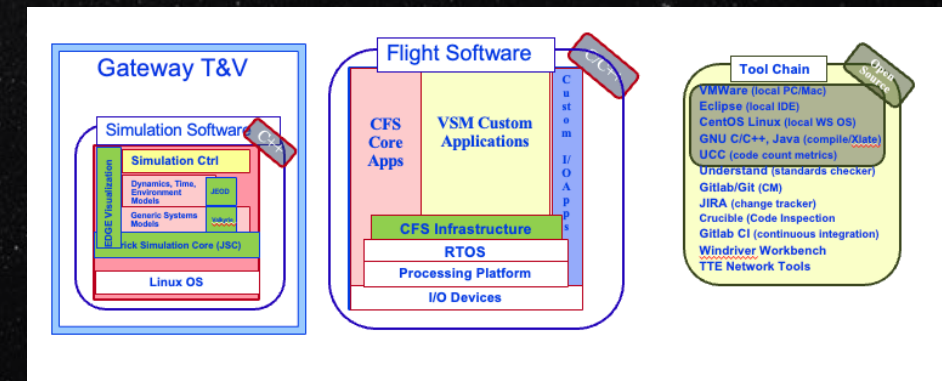
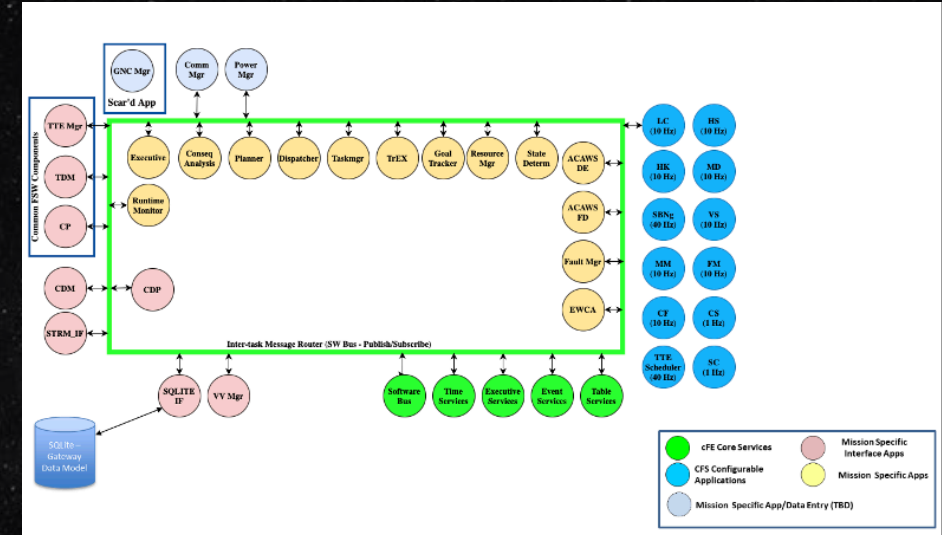
# JSC's Experience with core Flight System (cFS)

David Swartwout  
NASA / JSC / ER6



# NASA's core Flight System (cFS) Software Architecture

- cFS enables reuse, rapid development, and portability through its dynamic run-time environment, layered architecture, and component-based design
- Its three main components – the platform support package, operating system abstraction layer, and the core flight executive – give system designers the tools and flexibility they need to implement a robust FSW software that has powered 40+ small to large class NASA missions including the Roman Space Telescope, and is the primary software architecture for the Lunar Gateway being developed
- Developed originally by teams at GSFC, there is now a large user base of the open-source releases





# The cFS flight software framework takes advantage of a rich heritage of successful NASA flight software expertise.



## Write Once, Run Anywhere

Once you develop a cFS application you can easily port it to other hardware platforms and operating systems thanks to the Operating System abstraction Layer ([OSAL](#)) and the Platform Support Package ([PSP](#))



## Chat With The Experts

cFS has a solid and growing community of flight software engineers who are invested in improving the system and will answer questions about developing apps, customizing the source code, and implementing your embedded software project. [Join the Discussion.](#)



## What makes cFS Different

cFS originated from the collective experiences of flight software engineers at NASA. The cFS Team is committed to delivering high-quality, robust software that stands up to the rigorous standards and testing needed to ensure the safety and success of high-profile space missions. NASA is sharing cFS with the world to improve the quality and accessibility of embedded software for flight applications worldwide.



## Simple, Powerful Architecture

The tried-and-true bus architecture of the executive combined with the layered approach of cFS' abstraction layers and the modular approach to the codebase gives the simplicity needed by new users and the flexibility required by highly custom projects without sacrificing the quality and reliability of the system.

# cFS Core Components

Core components form the foundation of the cFS architecture, working together to provide a flexible, portable, and reusable framework for spacecraft flight software development across various missions and hardware platforms.

---

## **core Flight Executive (cFE)**

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

## **Operating System Abstraction Layer (OSAL)**

OSAL provides a single Application Program Interface (API) to the core Flight Executive (cFE) regardless of the underlying real-time operating system.

## **Platform Support Package (PSP)**

PSP provides a single Application Program Interface (API) to underlying avionics hardware and board support package.



## cFS Apps

Apps are architectural components which provides missions with advanced functions that go beyond those of cFE. Mission functionalities may include command and data handling, guidance, navigation, control, onboard data processing, and much more.

### Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol App (CF)

CF provides the capability to transmit files to and receive files from the ground. Tables are used for flexibility in specifying directory priorities and configurations.

### Checksum App (CS)

CS performs memory integrity management by verifying the contents of critical flight memory regions. Unexpected changes in memory are reported to ground operators.

### Command Ingest Lab App (ci\_lab)

ci\_lab is a simple command uplink application that accepts CCSDS telecommand packets over a UDP/IP port, as a test interface to a CFS system in a lab environment.

### File Data Store App (DS)

DS is used for storing software bus messages in files. Another cFS application such as CF must be used in order to transfer the files created by DS from their onboard storage location to where they will be viewed and processed.

### File Manager App (FM)

FM provides onboard file system management services by processing ground commands for copying, moving, and renaming files; decompressing files; creating directories; deleting files and directories; and more.

### Housekeeping App (HK)

HK provides the ability to organize data from various packets into new packets in order to best utilize the telemetry bandwidth available for a mission.

### Health and Safety App (HS)

The HS app provides functionality for application monitoring, event monitoring, hardware watchdog servicing, execution counter reporting (optional), and CPU aliveness indication via universal asynchronous receiver-transmitter (UART).

### Limit Checker App (LC)

LC is responsible for monitoring telemetry values and then issuing messages and activating scripts in the event of threshold limits.

### Memory Dwell App (MD)

MD monitors memory addresses accessed by the CPU. MD is used for both debugging and monitoring unanticipated telemetry not previously defined in the system prior to deployment.

### Memory Manager App (MM)

MM is used for the loading and dumping of system memory. MM supports command parameters, files, and symbolic addressing

### Sample App (sample\_app)

sample\_app is a simple cFE application to help verify that the cFE build and runtime software is configured correctly.

### Scheduler Lab App (sch\_lab)

sch\_lab is a simple packet scheduler application with a one second resolution. sch\_lab is intended to send housekeeping requests and other periodic packets in a lab/test cFS system.

### Stored Command App (SC)

The SC app provides the ability to execute onboard absolute-time and relative-time command sequences. The technology offers a generic implementation that can be configured by a user to fit the needs of a specific mission.

### Telemetry Output Lab App (to\_lab)

to\_lab is a simple telemetry downlink application that sends Consultative Committee for Space Data System (CCSDS) telemetry packets over a User Datagram Protocol (UDP) / Internet Protocol (IP) port as a test interface to a cFS system in a lab environment.

### cFS App Generator

The cFS App Generator is a template designed to generate a starter cFS app. It is currently in development.

## cFS Tools

Tools are utilized to support the development of cFS by offering modern engineering practices, testing capabilities, and essential resources for building and maintaining the environment.

---

### **ELF to CFE Table Tool (elf2cfetbl)**

elf2cfetbl is a ground utility to convert Executable and Linkable Format (ELF) to core Flight Executive (cFE) binary tables for cFS.

### **Ground System Lab Tool (cFS-GroundSystem)**

cFS-GroundSystem allows users to send commands and receive telemetry.

### **Table CRC Tool (tblCRCTool)**

tblCRCTool is a ground utility to generate binary table Cyclic Redundancy Checks (CRC) for cFS.

## cFS Interfaces

Interfaces serve as essential connectors between different software components or systems, enabling them to communicate and work together seamlessly within the cFS environment.

---

### **External Code Interface (ECI)**

ECI is a software abstraction layer which allows the interfacing of externally-generated task/mission-specific code to the cFS via a generic set of wrapper code.

### **Software Bus Network (SBN)**

SBN facilitates two-way communication from the cFS Software Bus service to an external application.

### **Simulink Interface Layer (SIL)**

SIL is an extension of the Simulink Coder generation tool which allows it to generate code which is compatible with the cFS ECI (External Code Interface).



# Class A Certification Process

The cFS Test Framework (CTF) can help provide artifacts to be used for certification. For class A certification, safety-critical flight code, these are the following NASA requirements, standards, and processes:

---

## **NPR-7150.2C – Software Engineering Requirements**

NPR 7150.2 establishes the engineering requirements that apply to the complete software development life cycle, including software planning, development, testing, maintenance, retirement, operations, management, acquisition & assurance activities.

---

## **JSC EA-WI-35 – Software Project Management & Development**

JSC Engineering Directorate Work Instruction that establishes the processes & work product templates necessary for developing software products adhering to NPR 7150.2 requirements.

---

## **NASA-STD-8739.8A – Software Assurance & Software Safety Standards**

Monitored by the project's Safety & Mission Assurance (S&MA) representative.  
Performed by the Independent Verification & Validation (IV&V) organization.

---


## **Spacecraft Software Engineering Team (SSET) policies & plans**


Based on the Capability Maturity Model Integration (CMMI) v2.0 model for Maturity Level 3.


# Class A Certification Artifacts

Certificate artifacts provide clear and tangible evidence that comprehensive development, testing, and validation procedures have been rigorously implemented throughout the project lifecycle.

---


 **Requirement Traceability Matrix**  
Requirement to code, to test cases,  
to verification methods


 **Verification & Validation**  
Test tool, test procedure, test  
scripts & expected test results

 **Software Detailed Design (SDD)  
Document**


 **Peer Review Metric Reports**  
Requirements, design, code & tests

 **Analysis Reports**  
Static code analyses, coverage gap  
analyses, safety analyses, etc.

 **Software Requirement  
Specifications (SRS) Document**

 **Unit tests and Code Coverage**  
Test procedure, test code &  
expected test results

 **Developer's Guide**

 **Version Description Document  
(VDD)**  
Including list of changes & open  
defects



# JSC's cFS Expertise

- Since ~2010, cFS has been the default software architecture for both human and robotic missions developed by JSC's Spacecraft Software Engineering Branch
- We have worked closely with the open-source cFS development team at GSFC, contributing many updates and tools to the ecosystem
- Through ongoing Gateway partnerships, we are collaborating with other prime contractors on their use and development of cFS deployments
- Specific for Gateway, we have developed many mission unique applications that are deployed across the vehicle to ensure commonality and operability



# Benefits and Usage

- From project initiation to an executable straw-man software architecture in days
  - Allow for early and continual integration as software capability grows
- Write code once and run on multiple platforms
  - From developer's desktop, through hardware in the loop labs, to the spacecraft
- Large open-source community of support
- In work on many JSC programs
  - Orion, Gateway, EHP
    - Recommended software architecture in Artemis software interoperability specification
- Utilized across NASA and industry on active and past missions



# References

- <https://etd.gsfc.nasa.gov/capabilities/core-flight-system/>