cFE
Executive Service (ES)
Tutorial

OSK v3.1

Executive Services (ES) – Overview

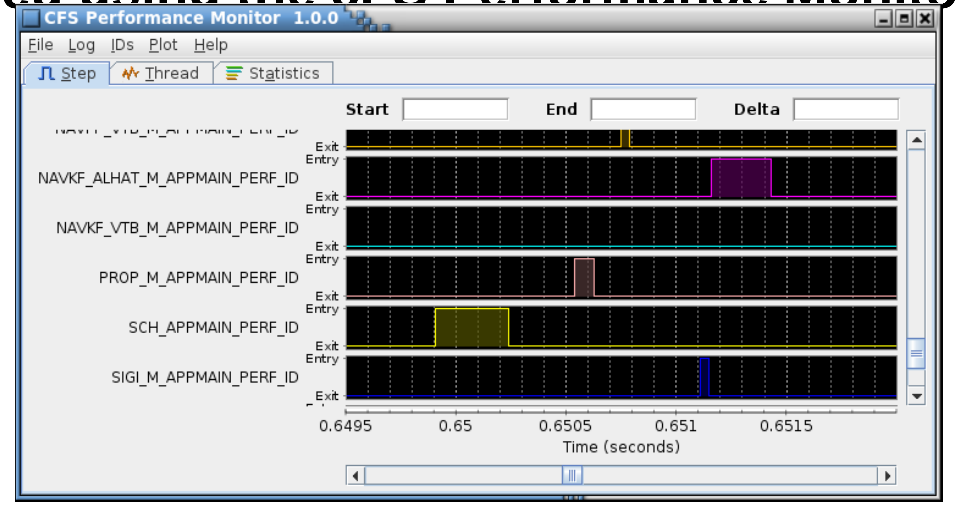
- **Initializes the cFE**
  - Reports reset type
  - Maintains an exception-reset log across processor resets
- **Creates the application runtime environment**
  - Primary interface to underlying operating system task services
  - Manages application resources
  - Supports starting, stopping, and loading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

Executive Services – Applications

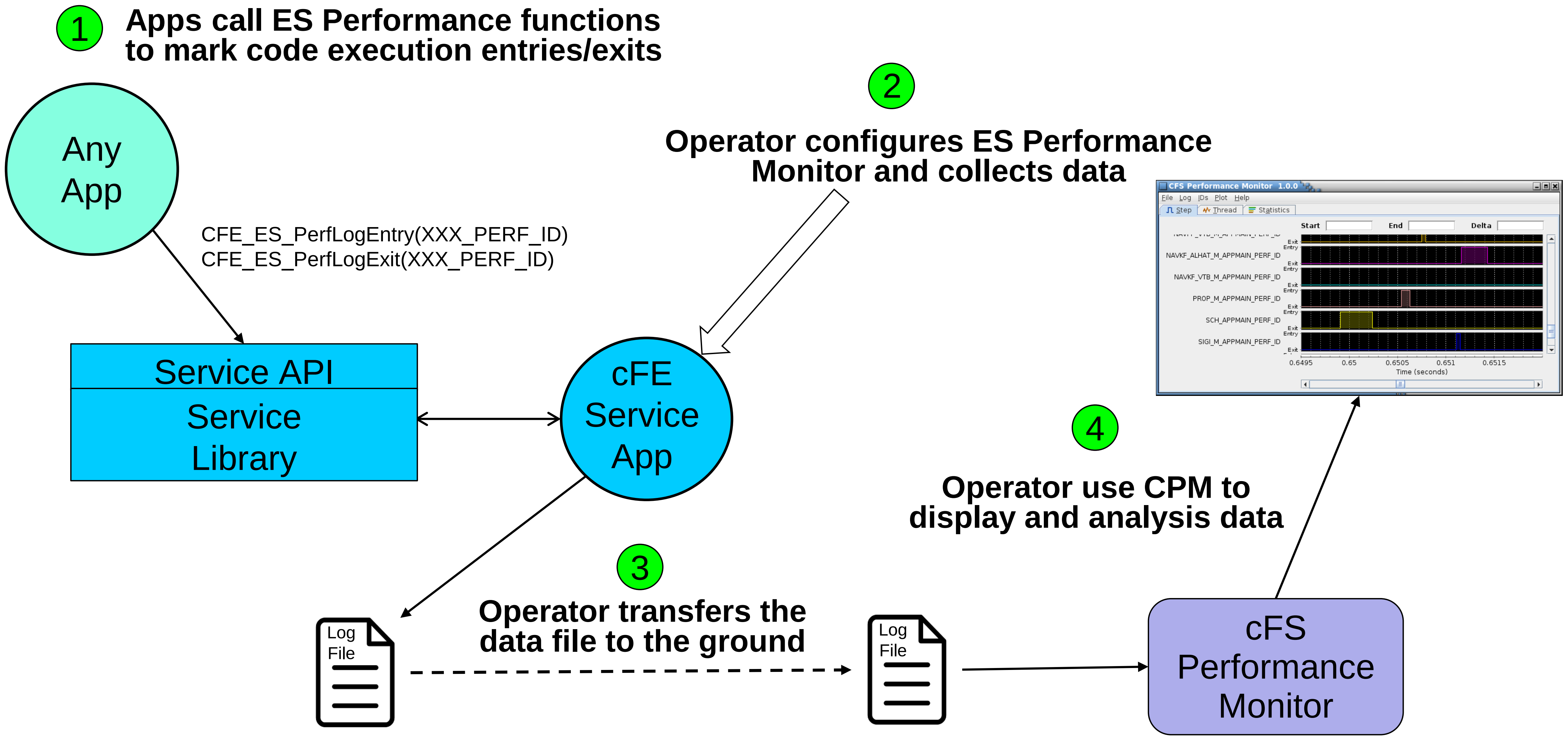
- **Applications are an architectural component that owns cFE and operating system resources**
- **Each application has a thread of execution in the underlying operating system (i.e. a task)**
- **Applications can create multiple child tasks**
  - Child tasks share the parent task’s address space
- **Mission applications are defined in *cfe_es_startup.scr* and loaded after the cFE applications are created**
- **Application Restarts and Reloads**
  - Start, Stop, Restart, Reload commands
  - Data is not preserved; application run through their initialization
  - Can be used in response to
    - Exceptions
    - On-board Failure Detection and Correction response
    - Ground commands

Executive Services – Performance Analyzer

- **Provides a method to identify and measure code execution paths**
  - System tuning, troubleshooting, CPU loading
- **Executive Service provides Developer inserts execution markers in FSW**
  - Entry marker indicate when execution resumes
  - Exit marker indicates when execution is suspended
  - CFE_ES_PerfLogExit() => CFE_SB_RcvMsg() => CFE_ES_PerfLogEntry()
- **Operator defines what markers should be captured via filters and defines triggers that determine when the filtered marker are captured**
- **Captured markers are written to a file that is transferred to the ground and displayed using the cFS Performance Monitor (CPM) tool**



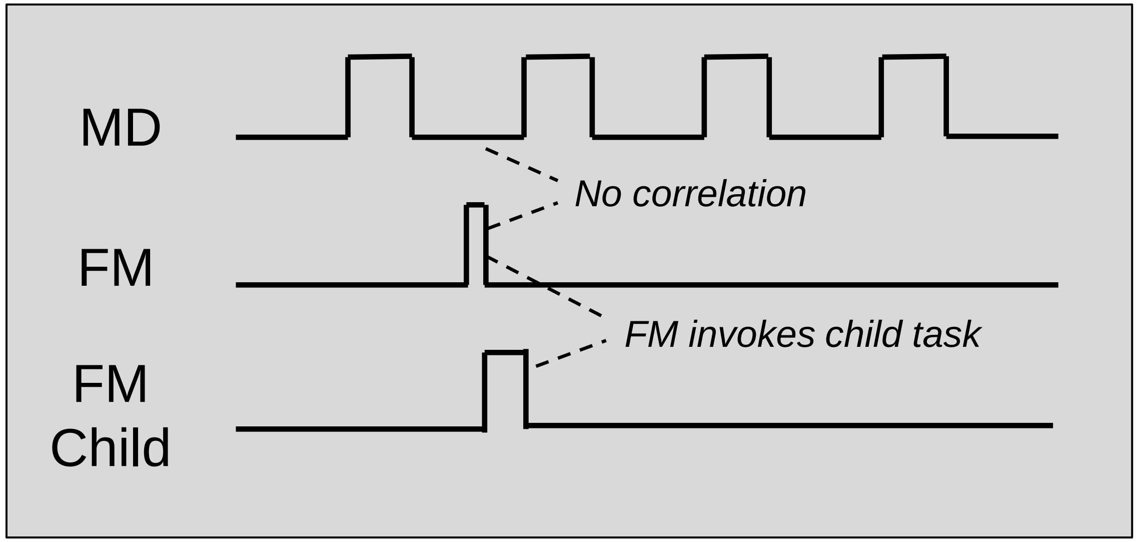
Performance Monitor Steps


1
Apps call ES Performance functions to mark code execution entries/exits
Any App
CFE_ES_PerfLogEntry(XXX_PERF_ID)
CFE_ES_PerfLogExit(XXX_PERF_ID)
Service API
Service Library
cFE Service App
2
Operator configures ES Performance Monitor and collects data
CFS Performance Monitor 1.0.0
File Log IDs Plot Help
Step Thread Statistics
Start End Delta
NAVKF_ALHAT_M_APPMAIN_PERF_ID
NAVKF_VTB_M_APPMAIN_PERF_ID
PROP_M_APPMAIN_PERF_ID
SCH_APPMAIN_PERF_ID
SIGI_M_APPMAIN_PERF_ID
Entry
Exit
0.6495
0.65
0.6505
0.651
0.6515
Time (seconds)
3
Operator transfers the data file to the ground
Log File
Log File
4
Operator use CPM to display and analysis data
cFS Performance Monitor

Performance Monitor Scenario

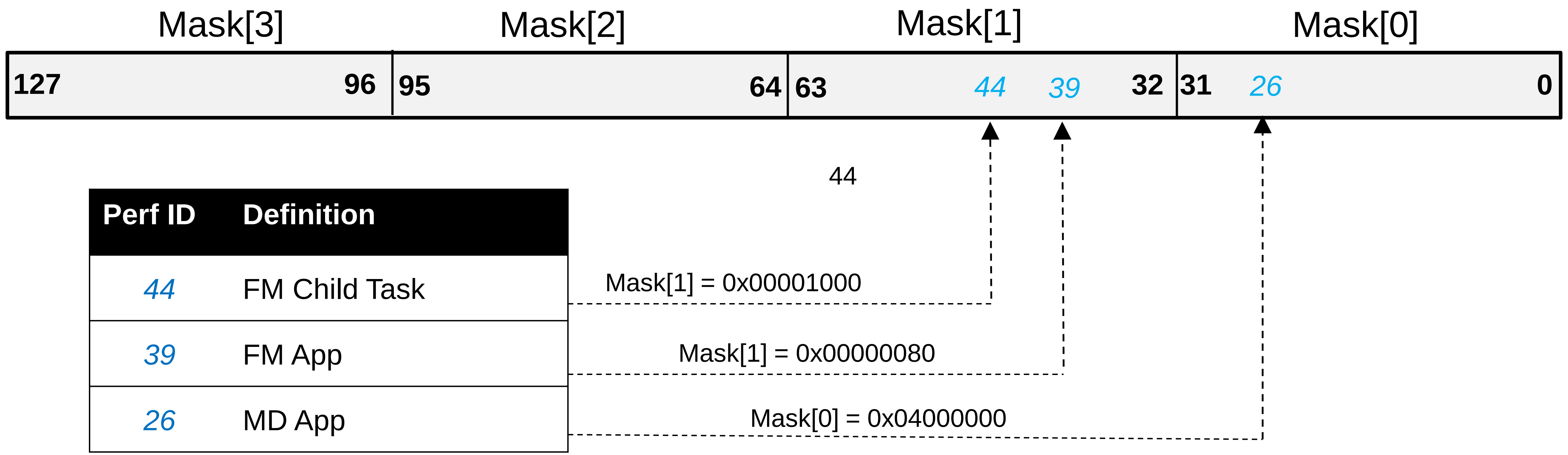
- **Trace execution of two apps**
  - File Manager: Pends indefinitely for a command to perform a file service
    - Uses a child task to perform services used in demo
  - Memory Dwell: Pends on 1Hz packet from Scheduler App
    - No correlation between MD's execution and FM's command processing.
  - "Send Housekeeping Packet Requests" from Scheduler App has been disabled for both apps
- **Data collection scenario**
  1. Start data collection
  2. Wait 4 seconds
  3. Issue FM "Send Directory in Telemetry" command
  4. Wait 4 seconds
  5. Issue FM "Write Directory to File" command
  6. Wait 4 seconds
  7. Issue FM "Send Directory in Telemetry" command
  8. Wait 4 seconds
  9. Stop data collection

Expected FM Command Trace



Performance Monitor Data Collection Configuration

- **Applications define performance IDs that must be unique within a platform**
  - Current convention is to define IDs in cfs/apps/xx/fsw/mission_inc/xx_perfids.h
- **A cFE ES command sets the "Filter Mask" that defines which performance IDs will be logged**
- **A cFE ES command sets the "Trigger Mask" that defines which performance IDs are used to start the data collection**
  - Data collection can also start when the ES collect data command is received
- **A "Masks" is an array of 32-bit words where each bit corresponds to a performance ID** ⚠

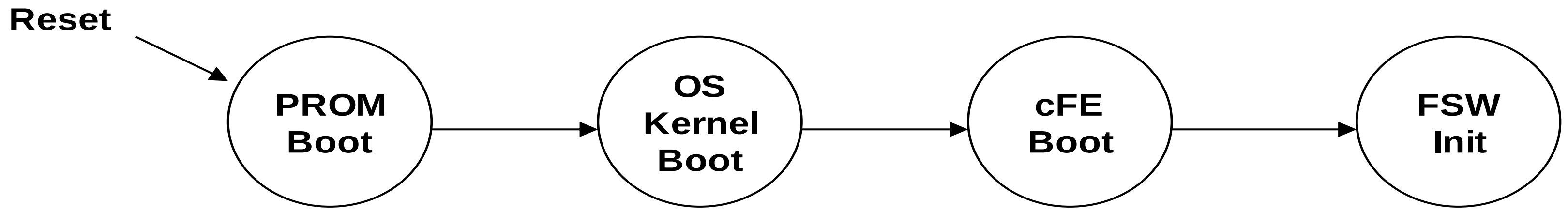


Perf ID	Definition
44	FM Child Task
39	FM App
26	MD App

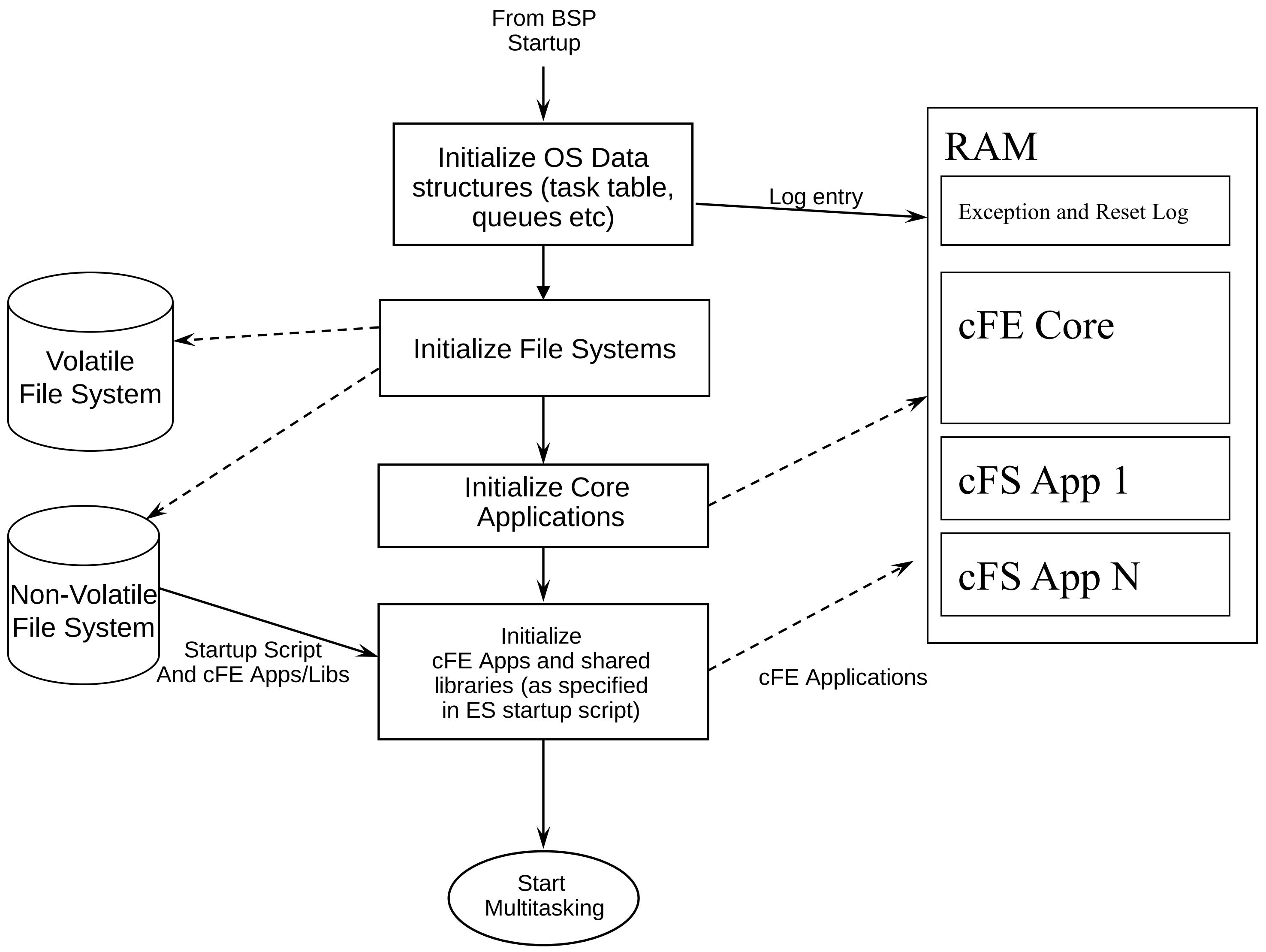
⚠ Requires knowledge of binary and hexadecimal numbering systems

Boot Sequence

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



cFE Executive Services Startup


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

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.

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

Example cfe_es_startup.scr

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

Executive Services – Logs

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

Executive Services – Reset Behavior

- **Power-on Reset**
  - Operating system loaded and started prior to cFE
  - Initializes file system
  - Critical data stores and logs cleared (initialized by hardware first)
  - ES starts each cFE service and then the mission applications
- **Processor Reset Preserves**
  - File system
  - Critical Data Store (CDS)
  - ES System Log
  - ES Exception and Reset (ER) log
  - Performance Analysis data
  - ES Reset info (i.e.reset type, boot source, number of processor resets)
  - Time Data (i.e. MET, SCTF, Leap Seconds)
- **A power-on reset will be performed after a configurable number of processor resets**
  - Ground responsible for managing processor reset counter
- **Service initialization order: ES, EVS, SB, TIME, TBL, FS**
- **Service app initialization order: EVS, SB, ES, TIME, TBL**
- **Applications images are often compressed in non-volatile memory and decompressed by ES when they are loaded**

Executive Services – Retrieving Onboard State

- **Telemetry**
  - Housekeeping Status
    - Log file states, App, Resets, Performance Monitor, Heap Stats

- **Telemetry packets generated by command**
  - Single App Information
  - Memory Pool Statistics Packet
  - Shell command output packet

- **Files generated by command**
  - System Log
  - Exception-Reset Log
  - Performance Monitor
  - Critical Data Store Registry
  - All registered apps
  - All registered tasks

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

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

OS	Call	Notes
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 – Configuration Parameters

- **List parameters that with higher probability of being tuned**

Executive Services APIs

Utility Functions	Purpose
CFE_ES_GetBlockInCDS	Allocate a block of space in the critical data store
CFE_ES_WriteToSysLog	Write to provided string to the System Log
CFE_ES_CalculateCRC	Calculate a data integrity value on a block of memory.

Critical Data Store (CDS) Functions	Purpose
CFE_ES_RegisterCDS	Allocates a block of memory in the Critical Data Store for a cFE Application
CFE_ES_CopyToCDS	Saves a block of data to the CDS
CFE_ES_RestoreFromCDS	Recover a block of data from the CDS

Memory Pool Functions	Purpose
CFE_ES_PoolCreate	Manages a memory pool created by an application
CFE_ES_GetPoolBuf	Gets a buffer from the memory pool created by CFE_ES_CreatePool
CFE_ES_PutPoolBuf	Releases a buffer from the memory pool

Performance Analysis Functions	Purpose
CFE_ES_PerfLogEntry	Entry marker for the performance analysis tool
CFE_ES_PerfLogExit	Exit marker for the performance analysis tool

Executive Services APIs

Application and Task Control Functions	Purpose
CFE_ES_GetResetType	Identifies the type of the last reset the processor most recently underwent
CFE_ES_ResetCFE	Perform a reset of the cFE Core and all of the cFE Applications
CFE_ES_RestartApp	Perform a restart of the specified cFE Application
CFE_ES_ReloadApp	Stops and then Starts a cFE Application from the specified file
CFE_ES_DeleteApp	Deletes a cFE Application
CFE_ES_ExitApp	Provides an exit point for a cFE Application’s run loop
CFE_ES_RegisterApp	Register the cFE Application
CFE_ES_GetAppIDByName	Returns the cFE Application ID corresponding to the given cFE Application name
CFE_ES_GetAppID	Returns the cFE Application ID of the calling cFE Application
CFE_ES_GetAppName	Returns the cFE Application Name of the calling cFE Application
CFE_ES_GetTaskInfo	Returns info about the specified child task ID including Task name, Parent task etc.
CFE_ES_RegisterChildTask	Register a child task (note each cFE Application has a main task)
CFE_ES_CreateChildTask	Create a child task
CFE_ES_DeleteChildTask	Delete a child task
CFE_ES_ExitChildTask	Exits a child task