Core Flight Executive Users Guide

Generated by Doxygen 1.8.13

Contents

2 Background

The Core Flight Executive (cFE) is an application development and run-time environment. The cFE provides a set of core services including Software Bus (messaging), Time, Event (Alerts), Executive (startup and runtime), and Table services. The cFE defines an application programming interface (API) for each service which serves as the basis for application development.

The cFE Software Bus service provides a publish and subscribe messaging system that allows applications to easily plug and play into the system. Applications subscribe to cFE services at runtime, making system modifications easy. Facilitating rapid prototyping, new applications can be compiled, linked, loaded, and started without requiring the entire system to be rebuilt.

Each service comes complete with a built in application that allows users to interface with each service. To support reuse and project independence, the cFE contains a configurable set of requirements and code. The configurable parameters allow the cFE to be tailored for each environment including desk-top and closed loop simulation environments. This provides the ability to run and test software applications on a developer's desktop and then deploy that same software without changes to the embedded system. In addition the cFE includes the following software development tools:

- Unit Test Framework (UTF) for unit testing applications developed via the cFE
- Software Timing Analyzer that provides visibility into the real-time performance of embedded systems software
- Table Builder
- Command and Telemetry utilities

The cFE is one of the components of the Core Flight System (cFS), a platform and project independent reusable software framework and set of reusable software applications. There are three key aspects to the cFS architecture: a dynamic run-time environment, layered software, and a component based design. The combination of these key aspects along with an implementation targeted to the embedded software domain makes it suitable for reuse on any number of NASA flight projects and/or embedded software systems.

The pivotal design feature, abstracting the software architecture from the hardware and forming the basis of reuse, is component layering. Each layer of the architecture "hides" its implementation and technology details from the other layers by defining and using standard Application Programming Interfaces (APIs). The internals of a layer can be changed without affecting other layers' internals and components.

The layers include an OS Abstraction Layer (OSAL), Platform Support Package (PSP) layer, core Flight Executive (cFE) layer, and an Application layer. The cFE layer runs on top of the PSP and OSAL layers. The cFE comes complete with a build environment, deployment guide, API reference guide, and provides a sample PSP. The OSAL is available open source and once integrated into the cFE build environment, developers will be ready to build and run the system and start developing their mission/project specific applications that easily plug and play into the system.

Core Flight Executive (cFE) Goals

The main long term goal of the cFE is to form the basis for a platform and project independent reusable software framework. The cFE with the OSAL allow the development of portable embedded system software that is independent of a particular Real Time Operating System and hardware platform. A secondary long term goal is to create a standardized, product-line approach for development of embedded aerospace flight software.

Functional and Community Goals

The cFE allows embedded system software to be developed and tested on desktop workstations and ported to the target platform without changing a single line of code, providing a shorter development and debug time. The cFE is an enabler of software collaboration amongst all users promoting the growth of the application and library layers where new applications, libraries, tools, and lessons learned can be contributed and shared.

It is important for application developers to realize the long term and functional goals of the cFE. With a standard set of services providing a standard API, all applications developed with the cFE have an opportunity to become useful on future missions through code reuse. In order to achieve this goal, applications must be written with care to ensure that their code does not have dependencies on specific hardware, software or compilers. The cFE and the underlying generic operating system API (OS API) have been designed to insulate the cFE Application developer from hardware and software dependencies. The developer, however, must make the effort to identify the proper methods through the cFE and OS API to satisfy their software requirements and not be tempted to take a "short-cut" and accomplish their goal with a direct hardware or operating system software interface.

3 Applicable Documents

Document ID	Document Title
MKS ID 112	cFE Requirements Document
582-2007-001	cFE Application Developers Guide
582-2007-001	cFE Application Developers Guide
582-2008-012	cFS Deployment Guide
	OS Abstraction Layer (OSAL) Library API
582-2007-00	OS Abstraction Layer (OSAL) Configuraion Guide

4 Version Numbers

Version Number Semantics

The version number is a sequence of four numbers, generally separated by dots when written. These are, in order, the Major number, the Minor number, the Implementation Revision number, and the Mission Revision number. At their option, Missions may modify the Mission Revision information as needed to suit their needs.

The Major number shall be incremented to indicate when there is a change to an API. Specifically when the API changes in ways that will cause existing correctly-written cFS components to stop working. It may also be incremented for a release that contains changes deemed to be of similar impact, even if there are no actual changes to the API.

The Minor number shall be incremented to indicate the addition of features to the API, which do not break existing code. It may also be incremented for a release that contains changes deemed to be of similar impact, even if there are no actual updates to the API.

The Implementation Version number shall be incremented when updates are made to an implementation that do not have consequences visible to external components. It may also be updated if there are other changes contained within a release that make it desirable for applications to distinguish one release from another.

The Mission Version number shall be set to zero in all officially released packages, and is entirely reserved for the use of the mission.

Version Number Flexibility

The major number may increment when there is no breaking change to the API, if the changes are significant enough to warrant the same level of attention as a breaking API change.

The minor number may increment when there have been no augmentations to the API, if changes are as significant as additions to the public API.

The revision numbers may update in implementations where no actual implementation-specific code has changed, if there are other changes within the release with similar significance.

How and Where Defined

The Major, Minor, and Revision components of the version are provided as simple macros defined in the cfe_version.h header file as part of the API definition; these macros must expand to simple integer values, so that they can be used in simple #if directives by the macro preprocessor.

The Mission Version is provided as a simple macro defined in the cfe_platform_cfg.h header file. As delivered in official releases, these macros must expand to simple integer values, so that they can be used in simple macro preprocessor conditions, but delivered code should not prevent a mission from, for example, deciding that the Mission Version is actually a text string.

5 Dependencies

The Core Flight Executive (cFE) is required to be built with the Operating System Abstraction Layer (OSAL) and Platform Support Package (PSP) componenets of the Core Flight System (cFS). It is always recommended to build with the latest versions of each of the components as backward compatability may not be supported.

Several internal data structures within the cFE use the "char" data type. This data type is typically 1 byte in storage size with a value range -128 to 127 or 0 to 255. The size of the "char" data type and whether or not the type is signed or unsigned can change across platforms. The cFE assumes use of the "char" data type as an **8-bit type.**

6 Acronyms

Acronym	Description
AC	Attitude Control
ACE	Attitude Control Electronics
ACS	Attitude Control System
API	Application Programming Interface
APID	CCSDS Application ID
App	Application
CCSDS	Consultative Committee for Space Data Systems
CDH, C&DH	Command and Data Handling
cFE	core Flight Executive

cFS	core Flight System
CM	Configuration Management
CMD	Command
CPU	Central Processing Unit
EDAC	Error Detection and Correction
EEPROM	Electrically Erasable Programmable Read-Only Memory
ES	Executive Services
EVS	Event Services
FC	Function Code
FDC	Failure Detection and Correction
FSW	Flight Software
HW, H/W	Hardware
ICD	Interface Control Document
MET	Mission Elapsed Time
MID	Message ID
OS	Operating System
OSAL	Operating System Abstraction Layer
PID	Pipeline ID
PKT	Packet
PSP	Platform Support Package
RAM	Random-Access Memory
SB	Software Bus
SDO	Solar Dynamics Observatory
ST5	Space Technology Five
STCF	Spacecraft Time Correlation Factor
SW, S/W	Software
TAI	International Atomic Time
TBD	To Be Determined
TBL	Table Services
TID	Task ID
TIME	Time Services
TLM	Telemetry
UTC	Coordinated Universal Time

7 Glossary of Terms

Term	Definition	
Application (or App)	A set of data and functions that is treated as a single entity by the cFE. cFE resources are allocated on a per-Application basis. Applications are made up of a Main Task and zero or more Child Tasks.	

Application ID	A processor unique reference to an Application. **NOTE: This is different from a CCSDS Application ID which is referred to as an "APID."**	
Application Programmer's Interface (API)	A set of routines, protocols, and tools for building software applications	
Board Support Package (BSP)	A collection of user-provided facilities that interface an OS and the cFE with a specific hardware platform. The BSP is responsible for hardware initialization.	
Child Task	A separate thread of execution that is spawned by an Application's Main Task.	
Command	A Software Bus Message defined by the receiving Application. Commands can originate from other onboard Applications or from the ground.	
Core Flight Executive (cFE)	A runtime environment and a set of services for hosting FSW Applications	
Critical Data Store (CDS)	A collection of data that is not modified by the OS or cFE following a Processor Reset.	
Cyclic Redundancy Check	A polynomial based method for checking that a data set has remained unchanged from one time period to another.	
Developer	Anyone who is coding a cFE Application.	
Event Data	Data describing an Event that is supplied to the cFE Event Service. The cFE includes this data in an Event Message.	
Event Filter	A numeric value (bit mask) used to determine how frequently to output an application Event Message defined by its Event ID.	

Event Format Mode	Defines the Event Message Format downlink option: short or long. The short format is used when there is limited telemetry bandwidth and is binary. The long format is in ASCII and is used for logging to a Local Event Log and to an Event Message Port.		
Event ID	A numeric literal used to uniquely name an Application event.		
Event Type	A numeric literal used to identify the type of an Application event.	An event type may be CFE_EVS_DEBUG, CFE_EVS_INFORMATION,	CFE_EVS_ERROR, or CFE_EVS_CRITICAL.
Event Message	A data item used to notify the user and/or an external Application of a significant event. Event Messages include a timestamp of when the message was generated, a processor unique identifier, an Application ID, the Event Type (DEBUG,INFO,ERROR or CRITICAL), and Event Data. An Event Message can either be real-time or playback from a Local Event Log.		

8 cFE User's Guide

- General Information and Concepts
  - Inter-Application Messages
  - File Systems and cFE Files
- Executive Services (ES)
  - cFE Executive Services Overview
  - cFE Executive Services Commands

  - cFE Executive Services Telemetry
  - ES Event Message Reference
  - cFE Executive Services Configuration Parameters

- Events Services (EVS)
  - cFE Event Services Overview
  - cFE Event Services Commands
  - cFE Event Services Telemetry
  - EVS Event Message Reference
  - cFE Event Services Configuration Parameters

- Software Bus Services (SB)
  - cFE Software Bus Overview
  - cFE Software Bus Commands
  - cFE Software Bus Telemetry
  - SB Event Message Reference
  - cFE Software Bus Configuration Parameters

- Table Services (TBL)
  - cFE Table Services Overview
  - cFE Table Services Commands
  - cFE Table Services Telemetry
  - TBL Event Message Reference
  - cFE Table Services Configuration Parameters

- Time Services (TIME)

8.1 Inter-Application Messages

8.2 File Systems and cFE Files

8.3 cFE Executive Services Overview

Executive Services (ES) is one of the five core Flight Executive components. ES is the primary interface to the underlying Operating System, providing a high level interface to system control facilities. The ES component is responsible for starting up and restarting the cFE, starting up, shutting down, and restarting cFE Applications, logging errors and performance data, and providing a persistent memory store for cFE Applications.

The interfaces to the ES task include the Ground Interface (commands and telemetry) and the Application Programmer Interfaces (APIs). The ES task interfaces to the OS through the OS Abstraction Layer and interfaces to the BSP through the BSP Abstraction Layer.

The services provided by the ES task include the Software Reset Service, Application and Child Task Service, File System Service, Performance Data Collection Service, Critical Data Store Service, Memory Pool Service, System Log Service, Shell Command Service and Device Service. Each of these services provide a full range of features that have been used on past missions.

For additional detail on Executive Services, see the following sections:

  - "Task"
  - "Startup Script"
- Software Reset Service
  - Reset Types and Subtypes
  - Exception and Reset (ER) Log
- Application and Child Task Service
  - Starting an Application
  - Stopping an Application
  - Restarting an Application
  - Reloading an Application
  - Listing Current Applications
  - Listing Current Tasks
  - Loading Common Libraries
  - Loading Device Servers
- File System Service
  - Defining the File Systems
  - Initializing File Systems
  - Listing the File Systems
- Performance Data Collection Service
  - Performance Data Collection Trigger Masks
  - Performance Data Collection Modes
  - Starting to Collect Performance Data

8.3.1 Terminology

The following sections describe terminology that is very relevant to understanding the Executive Services:

Next: Software Reset Service
Up To: cFE Executive Services Overview

8.3.2 "Application" and "cFE Application"

Application

The term 'Application' as defined in the Glossary of Terms is *a set of data and functions that is treated as a single entity by the cFE. cFE resources are allocated on a per-Application basis. Applications are made up of a Main Task and zero or more Child Tasks.*

Application

A 'cFE Application' is an application that is external to the cFE and designed to interface to the cFE through the APIs. It is created through an entry in the "Startup Script" (with the 'Object Type' field set to CFE_APP) or by way of the CFE_ES_START_APP_CC ground command.

When referring to one of the five applications internal to the cFE (ES, EVS, SB, TIME or TBL), the term 'internal application', 'cFE internal application' or "cFE Core Application" should be used.

A listing of cFE applications can be acquired by using the CFE_ES_QUERY_ALL_CC ground command. This listing will include the cFE internal applications as well as cFE applications that are loaded and running.

Next: "Task"
Up To: Terminology

8.3.3 "Task"

A Task is a thread of execution in the operating system, often associated with a cFE Application. Each cFE Application has a Main task providing its CPU context, stack and other OS resources. In addition, each cFE Application can create multiple Child Tasks which are closely associated with the Parent Task and cFE Application.

In a traditional Real Time Operating System such as vxWorks, the cFE Application Main task and child tasks end up being mapped to these OS tasks in the same shared memory space. For example, a Stored Command cFE Application that consists of a cFE Main Task and 10 Relative Time Sequence Child Tasks would have 11 tasks on a vxWorks system. The only association between these tasks exists in the cFE.

In a memory protected process oriented Operating System, the intention is to have a cFE Application implemented as a memory protected process with its own virtual address space. In this Process Model, each cFE Child Task would be a thread in the parent Process, much like a Unix process with multiple threads. In this model, the Stored Command example with a cFE Main Task and 10 Relative Time Sequence Child Tasks would consist of a Unix Process and 10 pthreads, all under the same virtual address space.

Next: "Startup Script"
Prev: "Application" and "cFE Application"
Up To: Terminology

8.3.4 "Startup Script"

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

The fields in a single entry include:

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

Immediately after the cFE completes its initialization, the ES Application first looks for the volatile startup script. The location in the file system is defined by the cFE platform configuration parameter named CFE_ES_VOLATILE_STARTUP_FILE. This configuration parameter contains a path as well as a filename. If the file is found, ES begins to startup the applications that are listed in the file. If ES does not find the file, it attempts to open the CFE_ES_NONVOL_STARTUP_FILE.

If ES finds the volatile startup script, the attempt to open the nonvolatile startup script is bypassed.

Any errors encountered in the startup script processing are written to the System Log Service. The System Log Service may also contain positive acknowledge messages regarding the startup script processing.

Refer to the CFS Deployment Guide for more information regarding the startup script. The startup script delivered with the cFE (`cfe_es_startup.scr`) also has some detailed information about the fields and the settings.

Next: Software Reset Service
Prev: Starting an Application
Up To: Terminology

8.3.5 "Application" and "cFE Application"

Application

The term 'Application' as defined in the Glossary of Terms is *a set of data and functions that is treated as a single entity by the cFE. cFE resources are allocated on a per-Application basis. Applications are made up of a Main Task and zero or more Child Tasks.*

Application

A 'cFE Application' is an application that is external to the cFE and designed to interface to the cFE through the APIs. It is created through an entry in the "Startup Script" (with the 'Object Type' field set to CFE_APP) or by way of the CFE_ES_START_APP_CC ground command.

When referring to one of the five applications internal to the cFE (ES, EVS, SB, TIME or TBL), the term 'internal application', 'cFE internal application' or "cFE Core Application" should be used.

A listing of cFE applications can be acquired by using the CFE_ES_QUERY_ALL_CC ground command. This listing will include the cFE internal applications as well as cFE applications that are loaded and running.

Next: "Task"
Up To: Terminology

8.3.6 "Task"

A Task is a thread of execution in the operating system, often associated with a cFE Application. Each cFE Application has a Main task providing its CPU context, stack and other OS resources. In addition, each cFE Application can create multiple Child Tasks which are closely associated with the Parent Task and cFE Application.

In a traditional Real Time Operating System such as vxWorks, the cFE Application Main task and child tasks end up being mapped to these OS tasks in the same shared memory space. For example, a Stored Command cFE Application that consists of a cFE Main Task and 10 Relative Time Sequence Child Tasks would have 11 tasks on a vxWorks system. The only association between these tasks exists in the cFE.

In a memory protected process oriented Operating System, the intention is to have a cFE Application implemented as a memory protected process with its own virtual address space. In this Process Model, each cFE Child Task would be a thread in the parent Process, much like a Unix process with multiple threads. In this model, the Stored Command example with a cFE Main Task and 10 Relative Time Sequence Child Tasks would consist of a Unix Process and 10 pthreads, all under the same virtual address space.

Next: "Startup Script"
Prev: "Application" and "cFE Application"
Up To: Terminology

8.3.7 "Startup Script"

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

The fields in a single entry include:

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

Immediately after the cFE completes its initialization, the ES Application first looks for the volatile startup script. The location in the file system is defined by the cFE platform configuration parameter named CFE_ES_VOLATILE_STARTUP_FILE. This configuration parameter contains a path as well as a filename. If the file is found, ES begins to startup

the applications that are listed in the file. If ES does not find the file, it attempts to open the CFE_ES_NONVOL_STA↩ RTUP_FILE.

If ES finds the volatile startup script, the attempt to open the nonvolatile startup script is bypassed.

Any errors encountered in the startup script processing are written to the System Log Service. The System Log Service may also contain positive acknowledge messages regarding the startup script processing.

Refer to the CFS Deployment Guide for more information regarding the startup script. The startup script delivered with the cFE (`cfe_es_startup.scr`) also has some detailed information about the fields and the settings.

Next: Software Reset Service
Prev: Starting an Application
Up To: Terminology

8.3.8 Software Reset Service

The ES Software Reset Service provides a command to reset the cFE as well as resetting individual applications. Because applications are dependent on the cFE services, it is not possible to reset the cFE without affecting the applications. Therefore, a command to reset the cFE will also reset every application that is running at the time the command is received.

Also part of this service is the Exception and Reset (ER) Log, which has a command for dumping or clearing the log and telemetry to show the number of entries in the log. In addition to the ER log, the user may find information about the most recent reset in the ES task housekeeping telemetry.

The ES Software Reset Service also provides a command to set the maximum number of processor resets before ES issues a power-on reset. There is a corresponding 'processor resets' counter in ES housekeeping telemetry that may be reset through another ES command.

Next: Reset Types and Subtypes
Prev: Terminology
Up To: cFE Executive Services Overview

8.3.9 Reset Types and Subtypes

The Reset Type is sent to the ground in the ES housekeeping packet and tells how the current running version of the cFE was invoked. The possible Reset Types expected in the telemetry field are CFE_ES_POWERON_RESET and C↩ FE_ES_PROCESSOR_RESET. There is a third Reset Type defined in the ES code as CFE_ES_APP_RESTART which applies only to restarting an individual application and is covered in more detail in the section titled Application and Child Task Service.

The Reset Subtype is also sent in the ES housekeeping packet and gives more detail about the type of reset that started the execution of the current running version of the cFE. The possible Reset Subtypes are CFE_ES_POWER_CYCLE, CFE_ES_PUSH_BUTTON, CFE_ES_HW_SPECIAL_COMMAND, CFE_ES_HW_WATCHDOG, CFE_ES_RESET_↩ COMMAND, CFE_ES_EXCEPTION, CFE_ES_UNDEFINED_RESET, CFE_ES_HWDEBUG_RESET, CFE_ES_BA↩ NKSWITCH_RESET.

Next: Exception and Reset (ER) Log
Prev: Software Reset Service
Up To: cFE Executive Services Overview

8.3.10 Exception and Reset (ER) Log

The Exception and Reset Log contains detailed information about past resets and exceptions. To view the information the CFE_ES_WRITE_ER_LOG_CC command must be sent. This command will write the log to a binary file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter CFE_ES_DEFAULT_ER_LOG_FILE is used to specify the path and filename. Use the ground system to get the file and display the contents. There is also a command to clear the ER log, CFE_ES_CLEAR_ER_↩ LOG_CC.

The size of the ER log is defined by the platform configuration parameter CFE_ES_ER_LOG_ENTRIES This log is preserved after a processor reset and held in the ES reset area.

A count of the number of entries in the log is present in the ES housekeeping telemetry. This count can be used with the configuration parameter # CFE_ES_ER_LOG_ENTRIES to calculate the fullness of the log.

The information contained in a single log entry is defined by the structure CFE_ES_ERLog_t.

Next: Application and Child Task Service
Prev: Reset Types and Subtypes
Up To: cFE Executive Services Overview

8.3.11 Application and Child Task Service

The ES Application and Child Task Service provide the user with full control over starting and stopping applications as well as querying information regarding applications, tasks and library routines.

There is no command to start or stop a child task. Child tasks can be controlled (started, stopped or deleted) only by the parent application through an API call.

This service provides a way for the user to load a set of library routines, (via the startup script) without starting a corresponding task. See the section related to library routines for more detail.

The ES task maintains a counter for the number of registered applications, number of registered child tasks and the number of registered libraries in the ES housekeeping data.

Next: Starting an Application
Prev: Software Reset Service
Up To: cFE Executive Services Overview

8.3.12 Starting an Application

There are two ways to start an application, through the ground command CFE_ES_START_APP_CC or through the startup script. In either case, the object file must be loaded on board before the command is sent or before the startup script is executed. The startup script contains a list of applications and library routines to load and start immediately after the cFE finishes its startup sequence. The parameters in the command, match the elements of an entry in the startup script. See the cFE Deployment Guide for more information about starting applications by way of the startup script.

The format of the Start Application command, is defined in the structure CFE_ES_StartApp_t. The members of the structure include, application name, entry point, filename, stack size, load address, exception action and priority.

If the command fails for any reason, an error event will be sent stating the reason for the failure. There may be additional information in the system log that can be viewed by sending the ES command to dump the system log.

After starting an application, the ES task sends an informational event message displaying the application name, filename of the object and the application ID. The new application will then show up in the query list downloaded in response to the CFE_ES_QUERY_ALL_CC command.

Next: Stopping an Application
Up To: Application and Child Task Service

8.3.13 Stopping an Application

Stopping an application can be done through the ground command CFE_ES_STOP_APP_CC. This command will terminate the application execution and all child tasks created by the application, free the system resources that it allocated and delete the corresponding object file.

The process of stopping an application is done in a controlled manner when the application is properly using the return code from the call to the CFE_ES_RunLoop. When the application properly uses this function, the ES task starts a timer and (via the return code) tells the application to exit at its own convenience. This gives the application time to free its own resources and do any cleanup that may be required before terminating itself by calling CFE_ES_ExitApp. If the timer expires and the application still exists, then ES must 'kill' the application. When the application is killed, ES attempts to cleanup the applications resources as best it could. In this case there is no guarantee that all the system resources are properly released.

The format of the Stop Application command, is defined in the structure CFE_ES_AppNameCmd_t. The only parameter in the command is an application name.

If the command fails for any reason, an error event will be sent stating the reason for the failure. There may be additional information in the system log that can be viewed by sending the ES command to dump the system log.

After stopping an application, the ES task sends a debug message stating the name of the application. After executing the command, the application (or any resources it allocated) should no longer be listed in any cFE tables or files.

Next: Restarting an Application
Prev: Starting an Application
Up To: Application and Child Task Service

8.3.14 Restarting an Application

The CFE_ES_RESTART_APP_CC command is used to restart an application. This command stops and restarts an application using the parameters defined when the application was originally started, either through the startup script or by way of the CFE_ES_START_APP_CC command.

Next: Reloading an Application
Prev: Stopping an Application
Up To: Application and Child Task Service

8.3.15 Reloading an Application

The CFE_ES_RELOAD_APP_CC command is used to reload an application. This command stops the application, unloads the object file, loads the new object file specified in the command and starts the application again using the parameters defined when the application was originally started, either through the startup script or by way of the CFE_ES_START_APP_CC command.

Next: Listing Current Applications
Prev: Restarting an Application
Up To: Application and Child Task Service

8.3.16 Listing Current Applications

There are two options for receiving information about applications, the CFE_ES_QUERY_ONE_CC command can be used to get details about a single application. This command takes an application name as its only parameter and the application information is sent as a software bus packet that can be telemetered to the ground.

Or the CFE_ES_QUERY_ALL_CC command can be used to get information about all the applications that are currently registered with ES. This command writes the application data to a file and has a one parameter which specifies the path and filename of the output file.

For either command, the following Application information is made available:

- **Application ID** - The Application ID assigned by the cFE to the Application
- **Type Identifier** - Identifies whether the Application is a CORE App or an EXTERNAL App
- **Name** - The Application Name
- **Entry Point** - The symbolic name for the entry point into the Application
- **Filename** - The name of the file the Application was loaded from
- **Stack Size** - The number of bytes allocated for the Application's stack
- **Load Address** - The starting address of memory where the Application was loaded
- **Load Size** - The size, in bytes, of the Application when loaded into memory
- **Start Address** - The physical address that maps to the Entry Point
- **Exception Action** - A flag that identifies whether the the Processor should undergo a Restart or whether just the Application should restart upon an exception condition within the Application
- **Priority** - The assigned priority for the Application
- **Main Task ID** - The Task ID assigned to the main task associated with the Application
- **Main Task Name** - The name of the main task associated with the Application
- **Number of Child Tasks** - The number of child tasks spawned by the main task

For a description of the format in which this data is dumped, see CFE_ES_AppInfo_t.

Next: Listing Current Tasks
Prev: Reloading an Application
Up To: Application and Child Task Service

8.3.17 Listing Current Tasks

The CFE_ES_QUERY_ALL_TASKS_CC command is used to get a list of child tasks that are currently registered with ES. The following information is provided for each registered task:

- **Task ID** - The Task ID associated with the specified task
- **Task Name** - The name of the Task
- **Application ID** - The ID for the Application the Task is associated with
- **Application Name** - The name of the Application the Task is associated with

Next: Loading Common Libraries
Prev: Listing Current Applications
Up To: Application and Child Task Service

8.3.18 Loading Common Libraries

Library routines may be loaded only through the startup script. There is an option that allows a library routine initialization function to be executed after the library is loaded. Refer to the cFE Application Developers Guide for more information regarding Library Routines and startup scripts. The startup script delivered with the cFE (`cfe_es_startup.scr`) also has some detailed information about library routines.

Next: Loading Device Servers
Prev: Listing Current Tasks
Up To: Application and Child Task Service

8.3.19 Loading Device Servers

Device Servers have been designed and coded but, as of Build 5.0 of the cFE, have not been tested.

Next: File System Service
Prev: Loading Common Libraries
Up To: Application and Child Task Service

8.3.20 File System Service

Next: Defining the File Systems
Prev: Application and Child Task Service
Up To: cFE Executive Services Overview

8.3.21 Defining the File Systems

Next: Initializing File Systems
Up To: File System Service

8.3.22 Initializing File Systems

Next: Listing the File Systems
Prev: Defining the File Systems
Up To: File System Service

8.3.23 Listing the File Systems

Next: Performance Data Collection Service
Prev: Initializing File Systems
Up To: File System Service

8.3.24 Performance Data Collection Service

The Performance Data Collection Service is also known as the Software Timing Analyzer. This service provides precise timing information for each software application by showing the task execution time in a waveform style similar to the display on a logic analyzer. In fact, this service closely models the operation of a logic analyzer in the way it displays, triggers and filters the data collected. This service uses passive markers that are inserted by the development team at key points in the code. The basic operation is to start the data collection, wait some amount of time, then send the command to stop the data collection. When the stop command is received, the ES task writes all the data from the buffer to a file. The file can then be imported to the windows GUI tool that is delivered with every version of the cFE and displayed on a windows PC. The size of the buffer is configurable through the CFE_ES_PERF_DATA_BUFFER_SIZE platform configuration parameter.

For more detail refer to the *CFE_STA_Integration.doc* file and the *Software Timing Analyzer Users Guide.pdf*. document. Both files are located in the `cFE/tools/perfutils` directory.

Additional information follows:

- Performance Data Collection Trigger Masks
- Performance Data Collection Modes
- Starting to Collect Performance Data
- Stopping the Collection of Performance Data
- Viewing the Collection of Performance Data

Next: Performance Data Collection Modes
Prev: File System Service
Up To: cFE Executive Services Overview

8.3.25 Performance Data Collection Trigger Masks

The trigger mask is used to tell the Performance Data Collection tool, precisely when to start collecting the data. There is a bit in the trigger mask for every marker used in the code. After a start command is received, the trigger mask is read and dictates when the performance tool begins storing data in the buffer.

If the trigger mask is set to all zeros, then the collection will begin immediately after the start command and continue until a stop command is received. In this case the buffer behaves in a 'circular' manner.

Next: Performance Data Collection Modes
Up To: Performance Data Collection Service

8.3.26 Performance Data Collection Modes

The Performance Data Collection modes are similar to the modes of a logic analyzer. The user may select a trigger point and 'start' the trace at the trigger point, 'center' the trace at the trigger point or 'end' the trace at the trigger point. **Note: The current cFE version does not have a way to set this mode. The ES software for this version of the cFE always uses the mode to "start the trace at the trigger point". The cFE FSW DCR 7092 can be used to track the issue.**

In 'start' mode, with a non-zero trigger mask, the first entry in the buffer will be the trigger point. The buffer then begins receiving unfiltered entries. This continues until the buffer becomes full or until the stop command is received. The result is a buffer that contains the trigger point and events that occur after the trigger point.

In 'center' mode, regardless of the trigger point, the buffer begins receiving unfiltered entries when the start command is received. In this case the buffer behaves as a circular buffer. When the trigger point is detected, the collection continues until 'buffer size divided by 2' entries have been stored or a stop command is received. The result is a buffer that contains data leading up to the trigger point, the trigger point itself and data following the trigger point. If the trigger never occurs and the stop cmd is received, the number of entries written to the file will correspond to the ES housekeeping telemetry point named 'Data Count'.

In 'end' mode, regardless of the trigger point, the buffer begins receiving unfiltered entries when the start command is received. In this case the buffer behaves as a circular buffer. The collection stops immediately after the trigger point is detected. The result is a buffer that contains data leading up to the trigger point and the trigger point itself as the last item in the buffer. If the trigger never occurs and the stop cmd is received, the number of entries written to the file will correspond to the ES housekeeping telemetry point named 'Data Count'.

The 'no trigger' mode can be used to continuously collect data until the stop command is received. This is done by setting the trigger mask to all zeros, then sending the start command. The result is a buffer that contains data up to the stop command. The number of entries written to the file will correspond to the ES housekeeping telemetry point named 'DataCount'.

Next: Starting to Collect Performance Data
Prev: Performance Data Collection Trigger Masks
Up To: Performance Data Collection Service

8.3.27 Starting to Collect Performance Data

The CFE_ES_START_PERF_DATA_CC command is used to start the data collection process. The ES task sends a debug event when the command is received. It is not possible to start a collection if the buffer-to-file write is in process from an earlier collection. There is an ES telemetry point that can be used to ensure there is not a buffer-to-file write in progress. This ES telemetry point is called 'Perf Data to Write' and begins counting down from 'Data Count' to zero. If this counter is zero, it is ok to send the start command. If any errors are encountered when the start command is received, the details will be displayed in an error event message.

Next: Stopping the Collection of Performance Data
Prev: Performance Data Collection Modes
Up To: Performance Data Collection Service

8.3.28 Stopping the Collection of Performance Data

The CFE_ES_STOP_PERF_DATA_CC command is used to stop the data collection process and write the buffer data to a file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter CFE_ES_DEFAULT_PERF_DUMP_FILENAME is used to specify the path and filename. The number of entries written to the file is determined by the 'data count' variable, which is sent in the ES housekeeping telemetry packet. To ensure cpu hogging does not occur during the write process, ES creates a low priority child task to perform the file write operation. This child task will write a number of entries, then sleep for a short time to give tasks of lower priority a chance to run. The number of entries between delays, and the delay time is displayed in the debug event at the time the stop command is received.

Next: Viewing the Collection of Performance Data
Prev: Performance Data Collection Modes
Up To: Performance Data Collection Service

8.3.29 Viewing the Collection of Performance Data

To view the performance data, the file created as a result of the stop command must be transferred to the ground and imported into the Software Timing Analyzer GUI tool. This tool is located in the `cFE/tools/perfutils` directory and designed only for Windows based machines. This GUI must be installed on the windows machine by way of the installation file named `SoftwareTimingAnalyzer.msi`. Double click this file on a windows machine to invoke the installation wizard. The installation process will place an icon named "*Timing Analyzer* on the desktop. Double click the icon and import the file. Refer to the *Software Timing Analyzer Users Guide.pdf*. document in the `cF↩E/tools/perfutils` directory for more information.

Next: Critical Data Store Service
Prev: Stopping the Collection of Performance Data
Up To: Performance Data Collection Service

8.3.30 Performance Data Collection Trigger Masks

The trigger mask is used to tell the Performance Data Collection tool, precisely when to start collecting the data. There is a bit in the trigger mask for every marker used in the code. After a start command is received, the trigger mask is read and dictates when the performance tool begins storing data in the buffer.

If the trigger mask is set to all zeros, then the collection will begin immediately after the start command and continue until a stop command is received. In this case the buffer behaves in a 'circular' manner.

Next: Performance Data Collection Modes
Up To: Performance Data Collection Service

8.3.31 Performance Data Collection Modes

The Performance Data Collection modes are similar to the modes of a logic analyzer. The user may select a trigger point and 'start' the trace at the trigger point, 'center' the trace at the trigger point or 'end' the trace at the trigger point. **Note: The current cFE version does not have a way to set this mode. The ES software for this version of the cFE always uses the mode to "start the trace at the trigger point". The cFE FSW DCR 7092 can be used to track the issue.**

In 'start' mode, with a non-zero trigger mask, the first entry in the buffer will be the trigger point. The buffer then begins receiving unfiltered entries. This continues until the buffer becomes full or until the stop command is received. The result is a buffer that contains the trigger point and events that occur after the trigger point.

In 'center' mode, regardless of the trigger point, the buffer begins receiving unfiltered entries when the start command is received. In this case the buffer behaves as a circular buffer. When the trigger point is detected, the collection continues until 'buffer size divided by 2' entries have been stored or a stop command is received. The result is a buffer that contains data leading up to the trigger point, the trigger point itself and data following the trigger point. If the trigger never occurs and the stop cmd is received, the number of entries written to the file will correspond to the ES housekeeping telemetry point named 'Data Count'.

In 'end' mode, regardless of the trigger point, the buffer begins receiving unfiltered entries when the start command is received. In this case the buffer behaves as a circular buffer. The collection stops immediately after the trigger point is detected. The result is a buffer that contains data leading up to the trigger point and the trigger point itself as the last item in the buffer. If the trigger never occurs and the stop cmd is received, the number of entries written to the file will correspond to the ES housekeeping telemetry point named 'Data Count'.

The 'no trigger' mode can be used to continuously collect data until the stop command is received. This is done by setting the trigger mask to all zeros, then sending the start command. The result is a buffer that contains data up to the stop command. The number of entries written to the file will correspond to the ES housekeeping telemetry point named 'DataCount'.

Next: Starting to Collect Performance Data
Prev: Performance Data Collection Trigger Masks
Up To: Performance Data Collection Service

8.3.32 Starting to Collect Performance Data

The CFE_ES_START_PERF_DATA_CC command is used to start the data collection process. The ES task sends a debug event when the command is received. It is not possible to start a collection if the buffer-to-file write is in process from an earlier collection. There is an ES telemetry point that can be used to ensure there is not a buffer-to-file write in progress. This ES telemetry point is called 'Perf Data to Write' and begins counting down from 'Data Count' to zero. If this counter is zero, it is ok to send the start command. If any errors are encountered when the start command is received, the details will be displayed in an error event message.

Next: Stopping the Collection of Performance Data
Prev: Performance Data Collection Modes
Up To: Performance Data Collection Service

8.3.33 Stopping the Collection of Performance Data

The CFE_ES_STOP_PERF_DATA_CC command is used to stop the data collection process and write the buffer data to a file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter CFE_ES_DEFAULT_PERF_DUMP_FILENAME is used to specify the path and filename. The number of entries written to the file is determined by the 'data count' variable, which is sent in the ES housekeeping telemetry packet. To ensure cpu hogging does not occur during the write process, ES creates a low priority child task to perform the file write operation. This child task will write a number of entries, then sleep for a short time to give tasks of lower priority a chance to run. The number of entries between delays, and the delay time is displayed in the debug event at the time the stop command is received.

Next: Viewing the Collection of Performance Data
Prev: Performance Data Collection Modes
Up To: Performance Data Collection Service

8.3.34 Viewing the Collection of Performance Data

To view the performance data, the file created as a result of the stop command must be transferred to the ground and imported into the Software Timing Analyzer GUI tool. This tool is located in the `cFE/tools/perfutils` directory and designed only for Windows based machines. This GUI must be installed on the windows machine by way of the installation file named `SoftwareTimingAnalyzer.msi`. Double click this file on a windows machine to invoke the installation wizard. The installation process will place an icon named "*Timing Analyzer* on the desktop. Double click the icon and import the file. Refer to the *Software Timing Analyzer Users Guide.pdf*. document in the `cF↩E/tools/perfutils` directory for more information.

Next: Critical Data Store Service
Prev: Stopping the Collection of Performance Data
Up To: Performance Data Collection Service

8.3.35 Critical Data Store Service

Some missions are required, for health, safety and mission success criteria, to survive Processor Resets. These mission requirements frequently flow down to Attitude Control and/or Command and Data Handling requirements that force an Application developer to design a mechanism for retaining software state information through a Processor Reset. The cFE provides the Critical Data Store service to assist the developer in meeting these requirements.

The Critical Data Store is an area of memory that is not cleared during a Processor Reset. In addition, the contents of memory are validated when accessed with a Data Integrity Value that helps to ensure the contents have not been corrupted. Each processor platform, through the design of its Board Support Package, can implement this area of memory in a number of ways to ensure the contents survive a Processor Reset. Applications can allocate a section of this memory for their use in a way similar to the cFE Table Services Overview.

When an Application registers a Critical Data Store (CDS), the Executive Services allocates a section of the Critical Data Store memory for the application's use and assigns the Application specified name to the memory area. The operator can find and learn the characteristics of these Critical Data Stores by using the Dump CDS Registry Command. This command will dump the contents of the CDS Registry maintained by the Executive Services into a file that can be downlinked and examined by the operator.

The CDS Registry dump will identify the following information for each registered CDS:

- **Handle** - the numeric identifier used by an Application to access the contents of the CDS
- **Size** - the number of bytes allocated to the specified CDS
- **Table Flag** - a flag that indicates whether the CDS is associated with a Critical Tables (when non-zero) or not (when equal to zero).
- **Name** - a processor specific name that uniquely identifies the CDS. The name comes in two parts, "`AppName`.↩ `CDSName`". `AppName` identifies which Application registered the CDS. `CDSName` is the name the Application assigned to the CDS.

The format of the CDS Registry Dump File is a cFE Standard File header (see CFE_FS_Header_t) followed by one or more CDS Registry Dump File Records (see CFE_ES_CDSRegDumpRec_t).

Next: Memory Pool Service
Prev: Performance Data Collection Service
Up To: cFE Executive Services Overview

8.3.36 Memory Pool Service

Refer to the cFE Application Developers Guide for additional information.

Applications that are designed for generic missions, frequently have to wait until run-time before allocating memory for buffers, data records, etc.

The cFE provides a memory allocation algorithm that may be used by an application to manage its block of memory. The user provides a pointer to its memory block and a list of block sizes and the cFE provides 'get' and 'put' API's to the user for managing its memory pool.

Run-time memory allocation in an embedded system can be risky because of the potential problem of memory fragmentation. Memory fragmentation is also referred to as External Fragmentation and is defined in the wikipedia as:

```
External fragmentation is the phenomenon in which free storage becomes divided
into many small pieces over time. It is a weakness of certain storage
allocation algorithms, occurring when an application allocates and deallocates
("frees") regions of storage of varying sizes, and the allocation algorithm
responds by leaving the allocated and deallocated regions interspersed. The
result is that, although free storage is available, it is effectively unusable
because it is divided into pieces that are too small to satisfy the demands of
the application. The term "external" refers to the fact that the unusable
storage is outside the allocated regions.
```

To help prevent this from happening, the cFE has integrated a memory allocation algorithm that is designed to create blocks at run-time, based on the size of the blocks requested. After a reset, there are no blocks created, the memory pool is said to be unconfigured. As requests for memory blocks are made, the memory pool first tries to use blocks that have been created but are no longer in use. If it cannot find an available block, it will create a new one. The created blocks remain until a reset occurs.

This algorithm is recommended when the size of the requests and the peak rate of requests can be pre-determined. It is highly recommended that adequate margin is designed into the pool size. The memory pool should never get close to being fully configured (i.e. not enough memory to create a new block). If the memory does become fully configured,

requests for new size blocks will fail, regardless of whether the created blocks are in-use or not. The margin on the memory pool can be monitored by viewing the 'free bytes' member of the memory pool statistics. The memory pool statistics are dumped only when commanded by way of the ES command CFE_ES_SEND_MEM_POOL_STATS_CC.

A user of the ES memory pool begins by tailoring the memory pool for the particular use, by defining a list of block sizes and allocating a block of memory. These block size definitions simply give the memory pool a set of sizes to choose from. They do not configure the memory pool in any way and they do not affect the size of the pool. The cFE defines a default set of block sizes in the cfe_platform_cfg.h file.

If the default block sizes are used, the application will create the pool using the simpler CFE_ES_PoolCreate API. This API takes a pointer to the first byte of the memory pool (allocated by the application) and a size parameter. The API returns a handle to be used for the get and put requests.

If the defaults are not sufficient, the user must define the block sizes and use the CFE_ES_PoolCreateEx API.

After receiving a positive response from the PoolCreate API, the memory pool is ready to accept requests, but at this point it is completely unconfigured (meaning there are no blocks created). The first valid request (via CFE_ES_Get↩ PoolBuf API) after creating the pool will always cause the memory pool to create a block and return a pointer to the new block. The size of the block depends on the size definitions mentioned earlier. If there is not an exact match between the requested and defined sizes, then the memory pool will create and return the smallest block that meets the following criteria: is a defined size and large enough to hold the request.

If another request for that size comes in before the first block was released through the CFE_ES_PutPoolBuf API, then the memory pool will create a second block of that size and return a pointer to the second block. If both blocks were then released through the CFE_ES_PutPoolBuf API and the memory pool statistics were dumped via the CFE_ES_↩ SEND_MEM_POOL_STATS_CC command, the number of blocks created would be two. The number of 'free bytes' in the pool would be the size of the pool minus the sum of the following items:

- the size of the two blocks created (even though they are not 'in-use').
- a buffer descriptor for each of the two blocks created ($2 * 12$ bytes)
- a 168 byte pool descriptor Refer to the cFE Applications Developers Guide for more details.

This allocation algorithm does have its limits. There are certain conditions that can place the memory pool in an undesired state. For instance, if a burst of get requests were received for the same block size, the memory pool may create a large number of blocks of that size. If this is a one-time burst, the memory pool would be configured with this large number of blocks that may no longer be needed. This scenario would use up the 'free bytes' margin in an undesired way. It should be noted that once the blocks are created, they cannot be deleted by any means other than a processor or power-on reset. It is highly recommended that the memory pool statistics be carefully monitored to ensure that the 'free-bytes' margin is sufficient (which is typically dictated by mission requirements).

An operator can obtain information about an Application's Memory Pool by using the Telemeter Memory Pool Statistics Command.

This command will cause Executive Services to extract pertinent statistics from the data used to manage the Memory Pool and telemeter them to the ground in the Memory Pool Statistics Telemetry Packet.

In order to obtain the statistics associated with a memory pool, the operator **MUST** have the correct Memory Handle as reported by the Application who owns the Memory Pool. **It should be noted that an inappropriate Memory Pool Handle can *(and likely will)* cause the system software to crash!** Within the cFE itself, there are three cFE Core Applications that make use of the Executive Services Memory Pool API. These are Software Bus (SB), Event Services (EVS) and Table Services (TBL). Each of these cFE Core Applications report their memory pool handles in telemetry.

The Memory Pool Statistics Telemetry Packet contains the following information:

- **Memory Pool Handle** - the handle, as provided by the operator in the Telemeter Memory Pool Statistics Command. This repeating of the handle in telemetry insures the operator knows which Memory Pool Statistics are being viewed

- **Pool Size** - The total size of the memory pool (in bytes)

- **Number Blocks Requested** - The total number of memory blocks requested for allocation

- **Number of Errors** - The total number of errors encountered when a block was released

- **Number of Free Bytes** - The total number of bytes in the Memory Pool that have never been allocated to a Memory Block

- **Block Statistics** - For each specified size of memory block (of which there are CFE_ES_MAX_MEMPOOL_BLOCK_SIZES), the following statistics are kept

  - **Block Size** - The size, in bytes, of all blocks of this type

  - **Number of Blocks Allocated** - The number of this sized block which are currently allocated and in use

  - **Number of Blocks Free** - The number of this size block which have been in use previously but are no longer being used

Next: System Log Service
Prev: Critical Data Store Service
Up To: cFE Executive Services Overview

8.3.37 System Log Service

The System Log is an array of bytes that contains back-to-back printf type messages from applications. The cFE internal applications use this log when errors are encountered during initialization before the Event Manager is fully initialized. To view the information the CFE_ES_WRITE_SYSLOG_CC command must be sent. This command will write the log to a binary file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter CFE_ES_DEFAULT_SYSLOG_FILE is used to specify the path and filename. Use the ground system to get the file and display the contents. The CFE_ES_CLEAR_SYSLOG_CC is used to clear the System log.

The size of the System log is defined by the platform configuration parameter CFE_ES_SYSTEM_LOG_SIZE. This log is preserved after a processor reset and held in the ES reset area.

A count of the number of entries in the log is present in the ES housekeeping telemetry.

Next: OS Shell Service
Prev: Memory Pool Service
Up To: cFE Executive Services Overview

8.3.38 OS Shell Service

Next: Version Identification Service
Prev: System Log Service
Up To: cFE Executive Services Overview

8.3.39 Version Identification Service

Next: Software Bus Frequently Asked Questions
Prev: OS Shell Service
Up To: cFE Executive Services Overview

8.3.40 Software Bus Frequently Asked Questions

Prev: OS Shell Service
Up To: cFE Executive Services Overview

8.4 cFE Executive Services Commands

The following is a list of commands that are processed by the cFE Executive Services Task.

Global CFE_ES_CLEAR_ER_LOG_CC
Clears the contents of the Exeception and Reset Log

Global CFE_ES_CLEAR_SYSLOG_CC
Clear Executive Services System Log

Global CFE_ES_DELETE_CDS_CC
Delete Critical Data Store

Global CFE_ES_DUMP_CDS_REGISTRY_CC
Dump Critical Data Store Registry to a File

Global CFE_ES_NOOP_CC
Executive Services No-Op

Global CFE_ES_OVER_WRITE_SYSLOG_CC
Set Executive Services System Log Mode to Discard/Overwrite

Global CFE_ES_QUERY_ALL_CC
Writes all Executive Services Information on All Applications to a File

Global CFE_ES_QUERY_ALL_TASKS_CC
Writes a list of All Executive Services Tasks to a File

Global CFE_ES_QUERY_ONE_CC
Request Executive Services Information on a Specified Application

Global CFE_ES_RELOAD_APP_CC
Stops, Unloads, Loads from a File and Restarts an Application

8.5 cFE Executive Services Telemetry

The following are telemetry packets generated by the cFE Executive Services Task.

8.6 cFE Executive Services Configuration Parameters

The following are configuration parameters used to configure the cFE Executive Services either for each platform or for a mission as a whole.

Global CFE_MISSION_ES_CDS_MAX_NAME_LEN
Maximum Length of Full CDS Name in messages

Global CFE_MISSION_ES_CDS_MAX_NAME_LENGTH
Maximum Length of CDS Name

Global CFE_MISSION_ES_DEFAULT_CRC
Mission Default CRC algorithm

Global CFE_MISSION_ES_MAX_APPLICATIONS
Mission Max Apps in a message

Global CFE_MISSION_ES_MAX_SHELL_CMD
Define Max Shell Command Size for messages

Global CFE_MISSION_ES_MAX_SHELL_PKT
Define Shell Command Telemetry Pkt Segment Size for messages

Global CFE_MISSION_ES_PERF_MAX_IDS
Define Max Number of Performance IDs for messages

Global CFE_MISSION_REV
Mission specific version number for cFE

Global CFE_PLATFORM_CORE_MAX_STARTUP_MSEC
CFE core application startup timeout

Global CFE_PLATFORM_ES_APP_KILL_TIMEOUT
Define ES Application Kill Timeout

Global CFE_PLATFORM_ES_APP_SCAN_RATE
Define ES Application Control Scan Rate

Global CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES
Define Maximum Number of Registered CDS Blocks

Global CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
Define ES Critical Data Store Memory Pool Block Sizes

Global CFE_PLATFORM_ES_CDS_SIZE
Define Critical Data Store Size

Global CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE
Default Application Information Filename

Global CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE
Default Critical Data Store Registry Filename

Global CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE
Default Exception and Reset (ER) Log Filename

Global CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME
Default Performance Data Filename

Global CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE
Define Max Size of Performance Data Buffer

Global CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS
Define Performance Analyzer Child Task Number of Entries Between Delay

Global CFE_PLATFORM_ES_PERF_FILTMASK_ALL
Define Filter Mask Setting for Enabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_FILTMASK_INIT
Define Default Filter Mask Setting for Performance Data Buffer

Global CFE_PLATFORM_ES_PERF_FILTMASK_NONE
Define Filter Mask Setting for Disabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_MAX_IDS
Define Max Number of Performance IDs

Global CFE_PLATFORM_ES_PERF_TRIGMASK_ALL
Define Filter Trigger Setting for Enabling All Performance Entries

Global CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
Define Default Filter Trigger Setting for Performance Data Buffer

Global CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
Define Default Filter Trigger Setting for Disabling All Performance Entries

Global CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING
RAM Disk Mount string

Global CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS
ES Ram Disk Number of Sectors

Global CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED
Percentage of Ram Disk Reserved for Decompressing Apps

Global CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE
ES Ram Disk Sector Size

Global CFE_PLATFORM_ES_RESET_AREA_SIZE
Define ES Reset Area Size

Global CFE_PLATFORM_ES_SHELL_OS_DELAY_MILLISEC
Define OS Task Delay Value for ES Shell Command

Global CFE_PLATFORM_ES_START_TASK_PRIORITY
Define ES Task Priority

Global CFE_PLATFORM_ES_START_TASK_STACK_SIZE
Define ES Task Stack Size

Global CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC
Startup script timeout

Global CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC
Poll timer for startup sync delay

Global CFE_PLATFORM_ES_SYSTEM_LOG_SIZE
Define Size of the cFE System Log.

Global CFE_PLATFORM_ES_USER_RESERVED_SIZE
Define User Reserved Memory Size

Global CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE
ES Volatile Startup Filename

Global CFE_PLATFORM_EVS_START_TASK_PRIORITY
Define EVS Task Priority

Global CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
Define EVS Task Stack Size

Global CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
Define SB Memory Pool Block Sizes

Global CFE_PLATFORM_SB_START_TASK_PRIORITY
Define SB Task Priority

Global CFE_PLATFORM_SB_START_TASK_STACK_SIZE
Define SB Task Stack Size

Global CFE_PLATFORM_TBL_START_TASK_PRIORITY
Define TBL Task Priority

Global CFE_PLATFORM_TBL_START_TASK_STACK_SIZE
Define TBL Task Stack Size

8.7 cFE Event Services Overview

Event Services (EVS) provides centralized control for the processing of event messages originating from the EVS task itself, other cFE core applications (ES, SB, TIME, and TBL), and from cFE applications. Event messages are asynchronous messages that are used to inform the operator of a significant event. EVS provides various ways to filter event messages in order to manage event message generation.

For more information on cFE Event Services, see the following sections:

- Event Message Format
- Local Event Log
- Event Message Control
- Event Message Filtering
- EVS Registry
- EVS Counters
- Resetting EVS Counters
- Effects of a Processor Reset on EVS
- Frequently Asked Questions about Event Services

8.7.1 Event Message Format

Event messages are software bus messages that contain the following fields:

- Application Name
- Event ID
- EventType
- Spacecraft ID
- Processor ID
- Message

The *Application Name* refers to the Application that issued the event message. The *Event ID* is an Application unique number that identifies the event. The Event Type can be one of four types: DEBUG, INFO, ERROR or CRITICAL. The *Spacecraft ID* and *Processor ID* identify the spacecraft and processor from which the event was generated. Note that the *Spacecraft ID* is defined in the cfe_mission_cfg.h file; The *Processor ID* is defined in the appropriate cfe_platform_cfg.h file. The *Message* is an ASCII text string describing the event. Event messages may have parameters associated with the event message. EVS formats the parameters such that they are part of the ASCII text string that make up the event message.

In order to accommodate missions that have limited telemetry bandwidth, EVS can be configured such that the ASCII text string part of the event message is omitted, thus reducing the size of each event message. This is referred to as *Short Format*; Event messages including the ASCII text string are referred to as *Long Format*. The default setting is specified in the cfe_platform_cfg.h file. EVS also provides commands in order to set the mode (short or long).

Since the design of the cFE's Software Bus is based on run-time registration, no predetermined message routing is defined, hence it is not truly correct to say that events are generated as telemetry. Technically, EVS generates events in the form of software bus messages. Applications such as Telemetry Output and Data Storage can then subscribe to these messages making them telemetry. For the purposes of this document, any references to telemetry assumes that a telemetry application subscribes to the EVS event software bus message and routes it to the ground as telemetry. Note that short format event messages on the Software Bus have different message lengths than long form messages and do not include any part of the long format message string.

The EVS can be configured via ground command to send event messages out one or more message ports. These message ports may include ports such as debug, console, and UART. Messages sent out of the message ports will be in ASCII text format. This is generally used for lab purposes. Note that the event mode (short or long) does affect the event message content sent out these message ports.

Next: Local Event Log
Up To: cFE Event Services Overview

8.7.2 Local Event Log

In addition to generating a software bus message, EVS logs the event message to a Local Event Log. Note that this is an optional feature that must be enabled via the cfe_platform_cfg.h file. The Local Event Log resides on the same processor as the EVS which is used to store events without relying on an external bus. In multi-processor cFE configurations the Local Event Buffer preserves event messages during non-deterministic processor initialization sequences and during failure scenarios. In order to obtain the contents of the Local Event Log, a command must be sent to write the contents of the buffer to a file which can then be sent to the ground via a file transfer mechanism. Note that event messages stored in the EVS Local Event Log are always long format messages and are not affected by the event mode (short or long).

EVS provides a command in order to clear the Local Event Log .

Local Event Log Mode

EVS can be configured to control the Local Event Log to either discard or overwrite the contents of the log when it becomes full. If the mode is set to overwrite, the log is treated like a circular buffer, overwriting the oldest event message contained in the log first. This control is configured by default in the cfe_platform_cfg.h file but can be modified by a command .

Next: Event Message Control
Prev: Event Message Format
Up To: cFE Event Services Overview

8.7.3 Event Message Control

In order for an application to be serviced by EVS, it must be registered with EVS. EVS provides various commands in order to control the event messages that are generated as software bus messages.

Event Message Control - By Type

The highest level of event message control that EVS provides is the ability to enable and disable event message types. As mentioned above, there are four event types. They are:

1. DEBUG
2. INFORMATION
3. ERROR
4. CRITICAL

When commands are sent to enable or disable a particular type of event message, ALL event messages of the specified type are affected. Typically, event meesages of type DEBUG are disabled on-orbit. Note that EVS provides the capability to affect multiple types within one command using a bit mask. Note also that the configuration parameter CFE_EV↩S_DEFAULT_TYPE_FLAG in the cfe_platform_cfg.h file specifies which event message types are enabled/disabled by default.

Event Message Control - By Application

Commands are available to enable and disable the generation of event messages for a particular application. The result is that ALL event messages for the specified Application are affected (i.e. enabled or disabled).

Event Message Control - By Event Type for an Application

EVS also provides the capability to enable / disable an event type for a particular application. Note that EVS provides the capability to affect multiple event types within one command using a bit mask.

Event Message Control - Individual Events

There are two ways to control the generation of individual events depending on whether the application's event message has been registered with EVS or not.

Modifying a registered event message filter

When an application registers with EVS, the application has the option of specifying the events that it wants to register for filtering along with the Event Message Filtering (only the Binary Filtering Scheme exists currently). Note that applications are limited in the number of events that they can register for filtering (see CFE_EVS_MAX_EVENT_FILTERS in cfe_platform_cfg.h for the mission defined limit). The filtering method uses a mask to determine if the message is forwarded to the software bus, making it available in telemetry (see Event Message Filtering for a description on filtering). Commands are available to modify the filter mask for any registered event.

An on-orbit mission, for example, might be experiencing a problem resulting in an application's event message being repeatedly issued, flooding the downlink. If the event message for the application is registered with EVS, then a command can be issued to set the event message filter to the specified value in order to prevent flooding of the downlink.

Adding/Removing an event message for filtering

Commands are also available to add filtering for those events that are not registered for filtering. Once an event is registered for filtering , the filter can be modified (see above) or removed .

An on-orbit mission, for example, might be experiencing a problem resulting in a event message being repeatedly issued, flooding the downlink. If the event message was not registered with EVS for filtering then the ground can add (i.e. register) the offending application's event for filtering (much like an application registers the event during initialization).

EVS also supports the ability to remove (i.e. unregister) an application's event message. Once it is removed, the event will no longer be filtered. Note that commands issued to disable events by event type, by application or by event type for an application are still valid and could affect this particular event.

Next: Event Message Filtering
Prev: Local Event Log
Up To: cFE Event Services Overview

8.7.4 Event Message Filtering

EVS uses a hexadecimal bit mask that controls how often a message is filtered. An event's filter mask is bit-wise ANDed with the event's event counter. There is one event counter for each event ID. If the result of the ANDing is zero then the message is sent.

Filter masks can be set so that one out of 1, 2, 4, 8 events are sent. Some examples of masks that use this pattern are: (0x0000, Every one), (0x0001, One of every 2), (0x0003, One of every 4), and (0x0007, One of every 8.

Filter masks can also be set so that only the first n events are sent. For example, the mask 0xFFFF generates one event message and then stops. Note that when the filter counter is reset to zero by command, this will restart the counting and enable n more events to be sent.

Event messages will be filtered until CFE_EVS_MAX_FILTER_COUNT events of the filtered event ID from the application have been received. After this, the filtering will become locked (no more of that event will be received by the ground) until the filter is either reset or deleted by ground command. This is to prevent the counter from rolling over, which would cause some filters to behave improperly. An event message will be sent when this maximum count is reached.

The following shows an example of how filtering works using a filter mask of x'0001', resulting in sending every other event:

	packet x	**packet X+1**	**packet X+2**	**packet X+3**	**packet X+4**	**...**
Event ID counter	x'0000'	x'0001'	x'0002'	x'0003'	x'0004'	
Event Filter mask	x'0001'	x'0001'	x'0001'	x'0001'	x'0001'	
Bitwise AND results	x'0000'	x'0001'	x'0000'	x'0001'	x'0000'	
Send event?	Yes	No	Yes	No	Yes	

In this example, the ground uses a filter mask of x'FFFE' resulting in the first two events being sent and then no more.

	packet x	**packet X+1**	**packet X+2**	**packet X+3**	**packet X+4**	**...**
Event ID counter	x'0000'	x'0001'	x'0002'	x'0003'	x'0004'	
Event Filter mask	x'FFFE'	x'FFFE'	x'FFFE'	x'FFFE'	x'FFFE'	
Bitwise AND results	x'0000'	x'0000'	x'0002'	x'0002'	x'0004'	
Send event?	Yes	Yes	No	No	No	

See cfe_evs.h for predefined macro values which can be used for masks.

Next: EVS Registry
Prev: Event Message Control
Up To: cFE Event Services Overview

8.7.5 EVS Registry

EVS maintains information on each registered application and all events registered for an application.

The registry contains the following information for each Registered Application:

- Active Flag - If equal to FALSE (0), all events from this Application are Filtered

- Event Count - Total number of events issued by this Application. Note that this value stop incrementing at 65535.

  The following information for each Filtered Event (up to CFE_EVS_MAX_EVENT_FILTERS):

- Event ID - Event ID for event whose filter has been defined
- Mask - Binary Filter mask value (see Event Message Filtering for an explanation)
- Count - Current number of times this Event ID has been issued by this Application

Next: EVS Counters
Prev: Event Message Filtering
Up To: cFE Event Services Overview

8.7.6 EVS Counters

There are 2 types of counters in EVS housekeeping telemetry:

- Total events sent counter
- Number of events sent for each Application

The difference is that the first one is the sum of all of the event messages sent. Both of these represent events that are actually sent (by EVS to the software bus). If an event message is filtered or disabled, neither counter is incremented.

There are other counters available that show how many event messages were generated by an App, however, these are only available for those events that are registered for filtering hence if you have a message that is not registered for filtering and the message type (e.g. DEBUG) is disabled then you won't know if the event was ever issued by an application. These counters are available by sending a command to write the EVS Application Data and transferring the file to the ground.

Next: Resetting EVS Counters
Prev: EVS Registry
Up To: cFE Event Services Overview

8.7.7 Resetting EVS Counters

As far as reset commands, there are 4 commands available:

1. Reset the total events sent counter
2. Reset the events sent counter for a particular Application - e.g. reset the LC application events counter
3. Reset all of the event counters for a particular registered event for a particular Application - e.g. Reset event counter for Event ID 5 for the LC Application.
4. Reset all of the event counters for ALL registered events for a particular App - e.g. Reset all registered event counters for LC.

Note that there is currently no way to reset ALL of the events sent counters for all of the Apps with one command.

Next: Effects of a Processor Reset on EVS
Prev: EVS Counters
Up To: cFE Event Services Overview

8.7.8 Effects of a Processor Reset on EVS

On a processor reset, the EVS Registry is cleared such that applications must re-register with EVS in order to use EVS services. All counters are also cleared with the exceptions of those listed below.

On a processor reset, the following EVS data is preserved (if the cFE is configured to include an Local Event Log):

- Local Event Log if the Local Event Log Mode is configured to Discard (1). If the Local Event Log Mode is configured to Overwrite (0), the contents of the log may be overwritten depending on the size and contents of the log prior to the reset.
- Local Event Log Full Flag
- Local Event Log overflow counter

The Local Event Log Mode (overwrite/discard) is set to the configured value specified in the cfe_platform_cfg.h file. The default value is Discard (1). Discard mode will guarantee the contents of the event log are preserved over a processor restart.

This provides the ground with the capability to write the Local Event Log to a file and transfer it to the ground in order to help debug a reset.

Next: Frequently Asked Questions about Event Services
Prev: Resetting EVS Counters
Up To: cFE Event Services Overview

8.7.9 Frequently Asked Questions about Event Services

(Q) My telemetry stream is being flooded with the same event message. How do I make it stop?
The most direct way to stop an event message from flooding your downlink stream is to send a command to EVS to filter the offending event (see Event Message Control or $sc_$cpu_EVS_SetBinFltrMask). In order to stop the event message from being sent, a bit mask of `'0xFFFF'` should be used. If the event is not currently registered for filtering, the event message must be added using the command $sc_$cpu_EVS_AddEvtFltr .
(Q) I filtered an event message and would now like to see it again. What do I do in order to see those events again?
If the event message that you are interested is registered with EVS for filtering, then you have 2 options: 1. You can use the $sc_$cpu_EVS_SetBinFltrMask command using a bit mask of `'0x0000'` which will result in getting all of the events for that Event Id **or** 2. You can remove the registration of that event with EVS (see $sc_$cpu_EVS_DelEvtFltr). Note that option (1) is the preferred method.
(Q) What is the purpose of DEBUG event messages?
Event message of type "DEBUG" are primarily used during flight software development in order to provide information that is most likely not needed on orbit. Some commands send debug event messages as verification that a command request was received. When writing the EVS local event log to a file, for example, an event message of type DEBUG is issued. On orbit, this event message is probably not needed. Instead, the command counter is used for command verification.
(Q) How do I find out which events are registered for filtering?
EVS provides a command ($sc_$cpu_EVS_WriteAppData2File) which generates a file containing all of the applications that have registered with EVS and all of the filters that are registered for each application. Note that EVS merely generates the file. The file must be transferred to the ground in order to view it.

Prev: Effects of a Processor Reset on EVS
Up To: cFE Event Services Overview

8.8 cFE Event Services Commands

The following is a list of commands that are processed by the cFE Event Services Task.

Global CFE_EVS_ADD_EVENT_FILTER_CC
Add Application Event Filter

Global CFE_EVS_CLEAR_LOG_CC
Clear Event Log

Global CFE_EVS_DELETE_EVENT_FILTER_CC
Delete Application Event Filter

Global CFE_EVS_DISABLE_APP_EVENT_TYPE_CC
Disable Application Event Type

8.9 cFE Event Services Telemetry

The following are telemetry packets generated by the cFE Event Services Task.

8.10 cFE Event Services Configuration Parameters

The following are configuration parameters used to configure the cFE Event Services either for each platform or for a mission as a whole.

Global CFE_MISSION_EVS_MAX_MESSAGE_LENGTH
Maximum Event Message Length

Global CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE
Default EVS Application Data Filename

Global CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
Default Event Log Filename

Global CFE_PLATFORM_EVS_DEFAULT_LOG_MODE
Default EVS Local Event Log Mode

Global CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE
Default EVS Message Format Mode

Global CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG
Default EVS Event Type Filter Mask

Global CFE_PLATFORM_EVS_LOG_MAX
Maximum Number of Events in EVS Local Event Log

Global CFE_PLATFORM_EVS_LOG_ON
Enable or Disable EVS Local Event Log

Global CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
Define Maximum Number of Event Filters per Application

Global CFE_PLATFORM_EVS_PORT_DEFAULT
Default EVS Output Port State

8.11 cFE Software Bus Overview

The Software Bus (SB) handles communication between software tasks on a processor. All tasks communicate with each other, with hardware devices, and with the ground by sending command and telemetry messages. The software bus provides an application programming interface (API) to other tasks for sending and receiving messages. This API is independent of the underlying operating system so that tasks can use the same interface regardless of which processor they reside on. Refer to the cFE Application Programmer's Interface (API) Reference for detailed information about the API functions.

The software bus is used internally by the flight software, and normally does not require attention from the ground. However, because of the scalability and the dynamic nature of the software bus, it is strongly recommended that each project carefully review the SB statistics and SB memory pool to be sure adequate margin is met on the configurable items.

The cFE software bus differs from earlier versions of the software bus in that it uses a dynamic protocol and builds its routing table at run-time through the SB subscribe API's. Also the cFE software bus pipes are created at run-time through the CFE_SB_CreatePipe API. Because the routing is established, and pipes are created at run-time, it is necessary to have a clear view of the routing details on command. The cFE software bus allows the user to dump the routing table, the pipe table, the message map and the statistics packet. Each of these items are described in detail in the corresponding section of this document.

8.11.1 Software Bus Terminology

In order to fully understand the Software Bus, it is imperative that the basic terms used to describe its features are also understood. Below are the critical terms that help identify what the Software Bus accomplishes for each Application:

- Messages
- Pipes
- Subscriptions
- Memory

Next: Messages
Up To: cFE Software Bus Overview

8.11.2 Messages

The sole purpose of the software bus is to provide applications a way to send messages to each other. The term message and the term packet are used interchangeably throughout this document. A message is a combined set of bytes with a predefined format that is used as the basis of communication on a spacecraft. All commands, telemetry, and other data that are passed between the ground and the spacecraft, and between subsystems of the spacecraft, are considered to be messages. The most common message format is CCSDS (Consultative Committee for Space Data Systems).

The cFE software bus was designed with 'hooks' to allow message formats other than CCSDS to be used. The APIs that are used to set and get message header fields are intentionally designed to be decoupled from CCSDS.

There are two general types of messages - commands (or command packets) and telemetry (or telemetry packets). Command packets are sent to a particular software task from the ground (or another task). Telemetry packets are sent from a particular software task to the ground (or other tasks).

Each packet begins with a header that includes the message identifier, often abbreviated as MsgId or message ID. The MsgId for CCSDS messages is the first 16 bits of the packet. The message 'type' indicator (command or telemetry) is embedded in the Message ID. The header also contains a packet length field and a packet sequence field. The packet sequence field is incremented by the software bus for telemetry packets each time a packet is sent. The software bus does not increment the sequence field for command packets. See the section named 'Packet Sequence Values' for more detail.

Telemetry packets typically contain a timestamp that indicates when the packet was produced. Command packets typically contain a command code that identifies the particular type of command.

The software bus provides APIs for 'setting' and 'getting' the fields in the header of the message.

Following the header is the user defined message data.

Next: Pipes
Up To: Software Bus Terminology

8.11.3 Pipes

The destinations to which messages are sent are called pipes. These are queues that can hold messages until they are read out and processed by a task. Each pipe is created at run-time through the CFE_SB_CreatePipe API. The pipe name and the pipe depth are given as arguments in the API. The pipe identifier (or PipeId) is given back to the caller after the API is executed. Each pipe can be read by only one task, but a task may read more than one pipe. Only the pipe owner is allowed to subscribe to messages on the pipe.

The Pipe IDs are specific to a particular processor (that is, the same ID number may refer to a different pipe on each processor). The pipe information for all pipes that have been created, may be requested at anytime by sending the 'Send Pipe Info' SB command . The software bus also provides a set of figures regarding capacity, current utilization and high water marks relevant to pipes. This information may be requested by sending the command to dump the SB statistics packet .

Next: Subscriptions
Prev: Messages
Up To: Software Bus Terminology

8.11.4 Subscriptions

A subscription is a run-time request for a particular message to be sent to a particular pipe. If the caller of the subscribe API is not the owner of the pipe, the request is rejected and an error event is sent. The application that creates the pipe is considered the owner of the pipe. The pipe specified in the subscription is sometimes referred to as the destination of the message. There are a maximum number of destinations for a particular message. This value is specified by the platform configuration parameter CFE_SB_MAX_DEST_PER_PKT.

As subscriptions are received, the destinations are added to the head of a linked list. During the sending of a message, the list is traversed beginning at the head of the list. Therefore the message will first be sent to the last subscriber. If an application has timing constraints and needs to receive a message in the shortest possible time, the developer may consider holding off its subscription until other applications have subscribed to the message.

The message limit specifies the maximum number of messages (with the specified Message ID) that are allowed on the specified pipe at any time. This limit is specified by the application at the time of the subscription. If the application uses the CFE_SB_Subscribe API, a message limit default value of four is used. If this default value is not sufficient, the caller would use the CFE_SB_SubscribeEx API that allows the message limit to be specified.

The software bus also provides the user with an option to unsubscribe to a message. The unsubscribe API takes two parameters, Message ID and Pipe ID. Only the owner of a pipe may unsubscribe to messages on that pipe.

Next: Memory
Prev: Pipes
Up To: Software Bus Terminology

8.11.5 Memory

The software bus statically allocates a block of memory for message buffers and subscription blocks. The size of this memory block is defined by the platform configuration parameter CFE_SB_BUF_MEMORY_BYTES. The memory is managed by the cFE ES memory pool and is used only by the software bus. The ES memory pool allows an application to define the block sizes for the pool at compile time. These sizes are defined by the platform configuration parameters prefixed with CFE_SB_MEM_BLOCK_SIZE (for example, CFE_SB_MEM_BLOCK_SIZE_01). It is recommended that a project tailor these values for the mission, based on the software bus packet sizes.

At the time a message is sent, two buffers are allocated from the pool. One for a buffer descriptor (CFE_SB_BufferD_t) and one for the size of the packet. Both buffers are returned to the pool when the message has been received by all recipients. More precisely, if there is one recipient for a message, the message buffers will be released on the following call to cFE_SB_RcvMsg for the pipe that received the message.

Also when subscriptions are received through the subscribe API's, the software bus allocates a subscription block (CFE_SB_DestinationD_t) from the pool. The subscription blocks are returned to the pool if and when the subscription is nullified through a CFE_SB_Unsubscribe call.

The software bus provides a set of figures regarding memory capacity, current memory utilization and high water marks relevant to the SB memory pool. This information may be requested by sending the command to dump the SB statistics packet. In addition, the current memory utilization value and the 'unmarked memory' value (CFE_SB_BUF_MEMORY_BYTES

- peak memory in use) are sent in software bus housekeeping telemetry. The unmarked memory value should be monitored regularly to ensure that the value (in bytes) does not continue to decline or approach zero. If this value were to approach zero, there is a possibility that memory requests would fail which may inhibit the sending of a message. The current memory utilization value should also be monitored to ensure the system contains no memory leaks. The value (in bytes) should remain stable under nominal conditions. Refer to the ES users guide for more information regarding the ES Memory Pool.

Next: Autonomous Actions
Prev: Subscriptions
Up To: Software Bus Terminology

8.11.6 Messages

The sole purpose of the software bus is to provide applications a way to send messages to each other. The term message and the term packet are used interchangeably throughout this document. A message is a combined set of bytes with a predefined format that is used as the basis of communication on a spacecraft. All commands, telemetry, and other data that are passed between the ground and the spacecraft, and between subsystems of the spacecraft, are considered to be messages. The most common message format is CCSDS (Consultative Committee for Space Data Systems).

The cFE software bus was designed with 'hooks' to allow message formats other than CCSDS to be used. The APIs that are used to set and get message header fields are intentionally designed to be decoupled from CCSDS.

There are two general types of messages - commands (or command packets) and telemetry (or telemetry packets). Command packets are sent to a particular software task from the ground (or another task). Telemetry packets are sent from a particular software task to the ground (or other tasks).

Each packet begins with a header that includes the message identifier, often abbreviated as MsgId or message ID. The MsgId for CCSDS messages is the first 16 bits of the packet. The message 'type' indicator (command or telemetry) is embedded in the Message ID. The header also contains a packet length field and a packet sequence field. The packet sequence field is incremented by the software bus for telemetry packets each time a packet is sent. The software bus does not increment the sequence field for command packets. See the section named 'Packet Sequence Values' for more detail.

Telemetry packets typically contain a timestamp that indicates when the packet was produced. Command packets typically contain a command code that identifies the particular type of command.

The software bus provides APIs for 'setting' and 'getting' the fields in the header of the message.

Following the header is the user defined message data.

Next: Pipes
Up To: Software Bus Terminology

8.11.7 Pipes

The destinations to which messages are sent are called pipes. These are queues that can hold messages until they are read out and processed by a task. Each pipe is created at run-time through the CFE_SB_CreatePipe API. The pipe name and the pipe depth are given as arguments in the API. The pipe identifier (or PipeId) is given back to the caller after the API is executed. Each pipe can be read by only one task, but a task may read more than one pipe. Only the pipe owner is allowed to subscribe to messages on the pipe.

The Pipe IDs are specific to a particular processor (that is, the same ID number may refer to a different pipe on each processor). The pipe information for all pipes that have been created, may be requested at anytime by sending the 'Send Pipe Info' SB command . The software bus also provides a set of figures regarding capacity, current utilization and high water marks relevant to pipes. This information may be requested by sending the command to dump the SB statistics packet .

Next: Subscriptions
Prev: Messages
Up To: Software Bus Terminology

8.11.8 Subscriptions

A subscription is a run-time request for a particular message to be sent to a particular pipe. If the caller of the subscribe API is not the owner of the pipe, the request is rejected and an error event is sent. The application that creates the pipe is considered the owner of the pipe. The pipe specified in the subscription is sometimes referred to as the destination of the message. There are a maximum number of destinations for a particular message. This value is specified by the platform configuration parameter CFE_SB_MAX_DEST_PER_PKT.

As subscriptions are received, the destinations are added to the head of a linked list. During the sending of a message, the list is traversed beginning at the head of the list. Therefore the message will first be sent to the last subscriber. If an application has timing constraints and needs to receive a message in the shortest possible time, the developer may consider holding off its subscription until other applications have subscribed to the message.

The message limit specifies the maximum number of messages (with the specified Message ID) that are allowed on the specified pipe at any time. This limit is specified by the application at the time of the subscription. If the application uses the CFE_SB_Subscribe API, a message limit default value of four is used. If this default value is not sufficient, the caller would use the CFE_SB_SubscribeEx API that allows the message limit to be specified.

The software bus also provides the user with an option to unsubscribe to a message. The unsubscribe API takes two parameters, Message ID and Pipe ID. Only the owner of a pipe may unsubscribe to messages on that pipe.

Next: Memory
Prev: Pipes
Up To: Software Bus Terminology

8.11.9 Memory

The software bus statically allocates a block of memory for message buffers and subscription blocks. The size of this memory block is defined by the platform configuration parameter CFE_SB_BUF_MEMORY_BYTES. The memory is managed by the cFE ES memory pool and is used only by the software bus. The ES memory pool allows an application to define the block sizes for the pool at compile time. These sizes are defined by the platform configuration parameters prefixed with CFE_SB_MEM_BLOCK_SIZE (for example, CFE_SB_MEM_BLOCK_SIZE_01). It is recommended that a project tailor these values for the mission, based on the software bus packet sizes.

At the time a message is sent, two buffers are allocated from the pool. One for a buffer descriptor (CFE_SB_BufferD_t) and one for the size of the packet. Both buffers are returned to the pool when the message has been received by all recipients. More precisely, if there is one recipient for a message, the message buffers will be released on the following call to cFE_SB_RcvMsg for the pipe that received the message.

Also when subscriptions are received through the subscribe API's, the software bus allocates a subscription block (CFE_SB_DestinationD_t) from the pool. The subscription blocks are returned to the pool if and when the subscription is nullified through a CFE_SB_Unsubscribe call.

The software bus provides a set of figures regarding memory capacity, current memory utilization and high water marks relevant to the SB memory pool. This information may be requested by sending the command to dump the SB statistics packet. In addition, the current memory utilization value and the 'unmarked memory' value (CFE_SB_BUF_MEMORY_BYTES

- peak memory in use) are sent in software bus housekeeping telemetry. The unmarked memory value should be monitored regularly to ensure that the value (in bytes) does not continue to decline or approach zero. If this value were to approach zero, there is a possibility that memory requests would fail which may inhibit the sending of a message. The current memory utilization value should also be monitored to ensure the system contains no memory leaks. The value (in bytes) should remain stable under nominal conditions. Refer to the ES users guide for more information regarding the ES Memory Pool.

Next: Autonomous Actions
Prev: Subscriptions
Up To: Software Bus Terminology

8.11.10 Autonomous Actions

The software bus is primarily a set of library routines that are called by other software tasks to send and receive packets. The software bus does not perform any operations autonomously, except for sending event messages if errors are detected during the transfer of packets.

As do other tasks, the SB task sends out housekeeping telemetry when requested through the 'Send Housekeeping Data' command.

Next: Operation of the SB Software
Prev: Software Bus Terminology
Up To: cFE Software Bus Overview

8.11.11 Operation of the SB Software

- Initialization
- All Resets
- Message Routing
- Packet Sequence Values
- Message Limit Error
- Pipe Overflow Error
- SB Event Filtering
- Diagnostic Data
- Control of Packet Routing
- Quality of Service
- Known Problem

Next: Initialization
Prev: Autonomous Actions
Up To: cFE Software Bus Overview

8.11.12 Initialization

No action is required by the ground to initialize the software bus. The software bus initializes internal data structures and tables the same way regardless of the type of reset.

Next: All Resets
Up To: Operation of the SB Software

8.11.13 All Resets

The software bus does not preserve any information across a reset of any kind. The software bus initializes internal data structures and tables the same way regardless of the type of reset. The routing is reestablished as the system initializes. It is normal procedure for each task of the system to create the pipe or pipes it needs and do all of its subscriptions during task initialization.

After any reset the following statements are true:

- The routing table is cleared and does not contain any routes.
- All subscriptions are lost and must be regenerated.
- The pipe table contains no data, all pipes must be recreated.
- Any packets in transit at the time of the reset are lost.
- The sequence counters for telemetry packets will begin again with a value of one.

Next: Message Routing
Prev: Initialization
Up To: Operation of the SB Software

8.11.14 Message Routing

In the software bus, all messages are processed in a similar way. The software bus uses the Message ID and the packet length fields (contained in the header) for routing the message to the destination pipe. If either of these two fields do not pass validation, the software bus generates an error event and aborts the delivery process. The software bus performs some validation checks by simply checking message header values against mission or platform configuration parameters. Messages originating from various tasks or instruments are routed to one or more pipes, where they wait until read by a task. The routing configuration for each message is established when applications call one of the SB subscribe APIs. The subscribe APIs take a Message ID and a Pipe ID as parameters. The routing for each packet is stored in SB memory and may be requested at any time by sending the 'Send Routing Info' command. The software bus also provides a set of figures regarding capacity, current utilization and high water marks relevant to the routing. This information may be requested by sending the command to dump the SB statistics packet.

Next: Packet Sequence Values
Prev: All Resets
Up To: Operation of the SB Software

8.11.15 Packet Sequence Values

The software bus populates the packet sequence header field for all telemetry messages that contain a current subscription. The first time a telemetry message with a new Message ID is sent, the sequence counter field in the header is set to a value of one. For subsequent sends of a message, the sequence counter is incremented by one regardless of the number of destinations for the packet. After a rollover condition the sequence counter will be a value of zero for one instance. The sequence counter is incremented in the CFE_SB_SendMsg API after all the checks have passed prior to the actual sending of the message. This includes the parameter checks, the 'no subscribers' check and the memory allocation check. Note: After passing all checks, the count is incremented regardless of whether the destinations are enabled or disabled.

Alternatively, the CFE_SB_PassMsg API can be used to pass a message without altering the sequence counter provided in the message. This will also not increment the sequence counter stored by the software bus. This method of message delivery is recommended for situations where the sender did not generate the packet, such as a network interface application passing a packet from a remote system to the local software bus.

The sequence counter for command messages is not altered by the software bus.

Next: Message Limit Error
Prev: Message Routing
Up To: Operation of the SB Software

8.11.16 Message Limit Error

Before placing a message on a pipe, the software bus checks the message limit to ensure the maximum number of packets in transit to the destination is not exceeded. If placing the message on the pipe would exceed the message limit, then the action of sending to that pipe is aborted and the 'Message Limit Error' event is sent. This condition will typically occur when an application that receives the packets does not respond quickly enough, or if the sender of the packets produces them too quickly.

This condition occurs often during development and during integration, for example when a remote processor gets reset or a 1553 cable becomes disconnected. Because of the common occurrences, the event may have filtering associated with it. Any filtering for this event would be performed by the cFE Event Services (EVS). Filtering for SB events may be specified in the cFE platform configuration file or may be commanded after the system initializes.

If this error occurs during nominal conditions, it could be an indication that the 'message limit' is not set correctly. The message limit is given at the time of the subscription and given as a parameter in the subscribe API. With the CFE_SB_Subscribe API, the SB uses a default message limit value specified by CFE_SB_DEFAULT_MSG_LIMIT. This constant is currently set to a value of four. If the default value is insufficient, the message limit value can be specified in the CFE_SB_SubscribeEx API.

A related failure is the pipe overflow condition, which can occur if the total number of packets (of all kinds) sent to a particular pipe is too large.

Next: Pipe Overflow Error
Prev: Packet Sequence Values
Up To: Operation of the SB Software

8.11.17 Pipe Overflow Error

Another common error that occurs during the send process is the pipe overflow error. This condition occurs if the total number of packets (of all kinds) sent to a particular pipe is too large. If this error occurs too frequently, it may be an indication that the pipe depth is not set correctly. The pipe depth is given at the time the pipe is created as a parameter in the CFE_SB_CreatePipe API.

Next: SB Event Filtering
Prev: Message Limit Error
Up To: Operation of the SB Software

8.11.18 SB Event Filtering

Most filtering for SB events is performed by the cFE Event Services (EVS). Filtering for SB events may be specified in the cFE platform configuration file or may be commanded after the system initializes. There is no SB event log that limits the number of events based on the capacity of the log, as in the heritage software bus.

There is one case in which events are filtered by the software bus instead of event services. This occurs when the software bus needs to suppress events so that a fatal recursive event condition does not transpire. Because the CFE_SB_SendMsg API is a library function that calls CFE_EVS_SendEvent, and CFE_EVS_SendEvent is a library function that calls CFE_SB_SendMsg, a calling sequence could cause a stack overflow if the recursion is not properly terminated. The cFE software bus detects this condition and properly terminates the recursion. This is done by using a set of flags (one flag per event in the Send API) which determine whether an API has relinquished its stack. If the CFE_SB_SendMsg needs to send an event that may cause recursion, the flag is set and the event is sent. CFE_EVS_SendEvent then calls CFE_SB_SendMsg in the same thread. If the second call to CFE_SB_SendMsg needs to send that same event again, it finds that the flag is set and the CFE_EVS_SendEvent call is bypassed, terminating the recursion. The result is that the user will see only one event instead of the many events that would normally occur without the protection. The heritage software bus did not have this condition because it stored events in the software bus event log and another thread would read them out at a later time.

Next: Diagnostic Data
Prev: Pipe Overflow Error
Up To: Operation of the SB Software

8.11.19 Diagnostic Data

The cFE software bus provides a set of commands to dump SB diagnostic data to help troubleshoot problems or check configuration settings. These commands allow the user to view the routing table, the pipe table or the message map. The message map is a lookup table used during a send operation to give fast access to the routing table index that corresponds to the message being sent.

The software bus also provides a statistics packet that can be used to tune the configuration parameters. This information is sent to the ground in the form of an SB packet when the corresponding command is received. The cFE limits the number of system pipes, unique Message IDs, buffer memory, messages on a pipe and subscriptions per Message ID. These limits are configurable through cFE platform and mission configuration parameters. The statistics packet was designed to let the project verify that these user settings provide the necessary margin to meet requirements.

The SB statistics information shows 'Currently In Use' figures, 'High Water Mark' figures and 'Max Allowed' figures for the following: buffer memory, messages on each pipe (pipe depth stats), System Pipes, Unique Message IDs and total subscriptions.

Depending on the task-scheduling implementation details of the operating system, it is possible to see the peak messages on a pipe occasionally exceed the depth of the pipe. The "Peak Messages In Use" parameter is included in the SB statistics packet under the pipe depth stats.

Next: Control of Packet Routing
Prev: SB Event Filtering
Up To: Operation of the SB Software

8.11.20 Control of Packet Routing

The software bus allows the ground to disable and enable the sending of packets of a specified Message ID to a specified pipe. All destinations that are needed for normal operation are enabled by default. Modifying the routing of packets may be required for the following reasons:

- In flight, one can enable diagnostic packets to see them on the ground.
- During testing, one can disable a destination to simulate an anomaly.

Next: Quality of Service
Prev: Diagnostic Data
Up To: Operation of the SB Software

8.11.21 Quality of Service

The software bus has a parameter in the CFE_SB_SubscribeEx API named Quality, which means Quality of Service (QOS) for off-board routing and is of the type CFE_SB_Qos_t. This structure has two members named priority and reliability. The Quality parameter is currently unused by the software bus. It is a placeholder to be used with the future software bus capability of inter-processor communication. Although currently the software bus does not read the Quality values, it would be best to set this parameter to the value defined as CFE_SB_Default_Qos. This value is set internally by the software bus with values of zero for priority and reliability. The values of zero will correspond to low priority and low reliability. Setting the QOS value to the CFE_SB_Default_Qos will ensure seamless integration when the software bus is expanded to support inter-processor communication.

Next: Known Problem
Prev: Control of Packet Routing
Up To: Operation of the SB Software

8.11.22 Known Problem

The software bus may perform unexpectedly under an unlikely corner-case scenario. This scenario was revealed in a stress test. The stress test was designed to deplete the Software Bus memory pool by having a high priority application continuously send 1000 byte packets to a lower priority application until the memory pool code returned an error code and sent the following event. "CFE_ES:getPoolBuf err:Request won't fit in remaining memory" At this point the higher priority sending application would stop executing. This would allow the lower priority receiving application to begin receiving the 1000 byte packets. After the receiving app processed all of the packets, the memory was restored to the memory pool as expected. The SB memory-in-use telemetry was zero because there were no software bus packets in transit. At this point any attempt to send a new-sized packet on the software bus was be rejected. The ES memory pool stated that the ". . . Request won't fit in remaining memory" even though there was currently no memory in use.

The simplest way to prevent this behavior is to ensure that there is margin when sizing the SB memory pool. To check the margin, monitor the "Peak Memory in Use" vs. the configuration parameter CFE_SB_BUF_MEMORY_BYTES which indicates the amount allocated.

Next: Frequently Asked Questions about Software Bus
Prev: Quality of Service
Up To: Operation of the SB Software

8.11.23 Initialization

No action is required by the ground to initialize the software bus. The software bus initializes internal data structures and tables the same way regardless of the type of reset.

Next: All Resets
Up To: Operation of the SB Software

8.11.24 All Resets

The software bus does not preserve any information across a reset of any kind. The software bus initializes internal data structures and tables the same way regardless of the type of reset. The routing is reestablished as the system initializes. It is normal procedure for each task of the system to create the pipe or pipes it needs and do all of its subscriptions during task initialization.

After any reset the following statements are true:

- The routing table is cleared and does not contain any routes.
- All subscriptions are lost and must be regenerated.
- The pipe table contains no data, all pipes must be recreated.
- Any packets in transit at the time of the reset are lost.
- The sequence counters for telemetry packets will begin again with a value of one.

Next: Message Routing
Prev: Initialization
Up To: Operation of the SB Software

8.11.25 Message Routing

In the software bus, all messages are processed in a similar way. The software bus uses the Message ID and the packet length fields (contained in the header) for routing the message to the destination pipe. If either of these two fields do not pass validation, the software bus generates an error event and aborts the delivery process. The software bus performs some validation checks by simply checking message header values against mission or platform configuration parameters. Messages originating from various tasks or instruments are routed to one or more pipes, where they wait until read by a task. The routing configuration for each message is established when applications call one of the SB subscribe APIs. The subscribe APIs take a Message ID and a Pipe ID as parameters. The routing for each packet is stored in SB memory and may be requested at any time by sending the 'Send Routing Info' command. The software bus also provides a set of figures regarding capacity, current utilization and high water marks relevant to the routing. This information may be requested by sending the command to dump the SB statistics packet.

Next: Packet Sequence Values
Prev: All Resets
Up To: Operation of the SB Software

8.11.26 Packet Sequence Values

The software bus populates the packet sequence header field for all telemetry messages that contain a current subscription. The first time a telemetry message with a new Message ID is sent, the sequence counter field in the header is set to a value of one. For subsequent sends of a message, the sequence counter is incremented by one regardless of the number of destinations for the packet. After a rollover condition the sequence counter will be a value of zero for one instance. The sequence counter is incremented in the CFE_SB_SendMsg API after all the checks have passed prior to the actual sending of the message. This includes the parameter checks, the 'no subscribers' check and the memory allocation check. Note: After passing all checks, the count is incremented regardless of whether the destinations are enabled or disabled.

Alternatively, the CFE_SB_PassMsg API can be used to pass a message without altering the sequence counter provided in the message. This will also not increment the sequence counter stored by the software bus. This method of message delivery is recommended for situations where the sender did not generate the packet, such as a network interface application passing a packet from a remote system to the local software bus.

The sequence counter for command messages is not altered by the software bus.

Next: Message Limit Error
Prev: Message Routing
Up To: Operation of the SB Software

8.11.27 Message Limit Error

Before placing a message on a pipe, the software bus checks the message limit to ensure the maximum number of packets in transit to the destination is not exceeded. If placing the message on the pipe would exceed the message limit, then the action of sending to that pipe is aborted and the 'Message Limit Error' event is sent. This condition will typically occur when an application that receives the packets does not respond quickly enough, or if the sender of the packets produces them too quickly.

This condition occurs often during development and during integration, for example when a remote processor gets reset or a 1553 cable becomes disconnected. Because of the common occurrences, the event may have filtering associated with it. Any filtering for this event would be performed by the cFE Event Services (EVS). Filtering for SB events may be specified in the cFE platform configuration file or may be commanded after the system initializes.

If this error occurs during nominal conditions, it could be an indication that the 'message limit' is not set correctly. The message limit is given at the time of the subscription and given as a parameter in the subscribe API. With the CFE_SB_Subscribe API, the SB uses a default message limit value specified by CFE_SB_DEFAULT_MSG_LIMIT. This constant is currently set to a value of four. If the default value is insufficient, the message limit value can be specified in the CFE_SB_SubscribeEx API.

A related failure is the pipe overflow condition, which can occur if the total number of packets (of all kinds) sent to a particular pipe is too large.

Next: Pipe Overflow Error
Prev: Packet Sequence Values
Up To: Operation of the SB Software

8.11.28 Pipe Overflow Error

Another common error that occurs during the send process is the pipe overflow error. This condition occurs if the total number of packets (of all kinds) sent to a particular pipe is too large. If this error occurs too frequently, it may be an indication that the pipe depth is not set correctly. The pipe depth is given at the time the pipe is created as a parameter in the CFE_SB_CreatePipe API.

Next: SB Event Filtering
Prev: Message Limit Error
Up To: Operation of the SB Software

8.11.29 SB Event Filtering

Most filtering for SB events is performed by the cFE Event Services (EVS). Filtering for SB events may be specified in the cFE platform configuration file or may be commanded after the system initializes. There is no SB event log that limits the number of events based on the capacity of the log, as in the heritage software bus.

There is one case in which events are filtered by the software bus instead of event services. This occurs when the software bus needs to suppress events so that a fatal recursive event condition does not transpire. Because the CFE_SB_SendMsg API is a library function that calls CFE_EVS_SendEvent, and CFE_EVS_SendEvent is a library function that calls CFE_SB_SendMsg, a calling sequence could cause a stack overflow if the recursion is not properly terminated. The cFE software bus detects this condition and properly terminates the recursion. This is done by using a set of flags (one flag per event in the Send API) which determine whether an API has relinquished its stack. If the CFE_SB_SendMsg needs to send an event that may cause recursion, the flag is set and the event is sent. CFE_EVS_SendEvent then calls CFE_SB_SendMsg in the same thread. If the second call to CFE_SB_SendMsg needs to send that same event again, it finds that the flag is set and the CFE_EVS_SendEvent call is bypassed, terminating the recursion. The result is that the user will see only one event instead of the many events that would normally occur without the protection. The heritage software bus did not have this condition because it stored events in the software bus event log and another thread would read them out at a later time.

Next: Diagnostic Data
Prev: Pipe Overflow Error
Up To: Operation of the SB Software

8.11.30 Diagnostic Data

The cFE software bus provides a set of commands to dump SB diagnostic data to help troubleshoot problems or check configuration settings. These commands allow the user to view the routing table, the pipe table or the message map. The message map is a lookup table used during a send operation to give fast access to the routing table index that corresponds to the message being sent.

The software bus also provides a statistics packet that can be used to tune the configuration parameters. This information is sent to the ground in the form of an SB packet when the corresponding command is received. The cFE limits the number of system pipes, unique Message IDs, buffer memory, messages on a pipe and subscriptions per Message ID. These limits are configurable through cFE platform and mission configuration parameters. The statistics packet was designed to let the project verify that these user settings provide the necessary margin to meet requirements.

The SB statistics information shows 'Currently In Use' figures, 'High Water Mark' figures and 'Max Allowed' figures for the following: buffer memory, messages on each pipe (pipe depth stats), System Pipes, Unique Message IDs and total subscriptions.

Depending on the task-scheduling implementation details of the operating system, it is possible to see the peak messages on a pipe occasionally exceed the depth of the pipe. The "Peak Messages In Use" parameter is included in the SB statistics packet under the pipe depth stats.

Next: Control of Packet Routing
Prev: SB Event Filtering
Up To: Operation of the SB Software

8.11.31 Control of Packet Routing

The software bus allows the ground to disable and enable the sending of packets of a specified Message ID to a specified pipe. All destinations that are needed for normal operation are enabled by default. Modifying the routing of packets may be required for the following reasons:

- In flight, one can enable diagnostic packets to see them on the ground.
- During testing, one can disable a destination to simulate an anomaly.

Next: Quality of Service
Prev: Diagnostic Data
Up To: Operation of the SB Software

8.11.32 Quality of Service

The software bus has a parameter in the CFE_SB_SubscribeEx API named Quality, which means Quality of Service (QOS) for off-board routing and is of the type CFE_SB_Qos_t. This structure has two members named priority and reliability. The Quality parameter is currently unused by the software bus. It is a placeholder to be used with the future software bus capability of inter-processor communication. Although currently the software bus does not read the Quality values, it would be best to set this parameter to the value defined as CFE_SB_Default_Qos. This value is set internally by the software bus with values of zero for priority and reliability. The values of zero will correspond to low priority and low reliability. Setting the QOS value to the CFE_SB_Default_Qos will ensure seamless integration when the software bus is expanded to support inter-processor communication.

Next: Known Problem
Prev: Control of Packet Routing
Up To: Operation of the SB Software

8.11.33 Known Problem

The software bus may perform unexpectedly under an unlikely corner-case scenario. This scenario was revealed in a stress test. The stress test was designed to deplete the Software Bus memory pool by having a high priority application continuously send 1000 byte packets to a lower priority application until the memory pool code returned an error code and sent the following event. "CFE_ES:getPoolBuf err:Request won't fit in remaining memory" At this point the higher priority sending application would stop executing. This would allow the lower priority receiving application to begin receiving the 1000 byte packets. After the receiving app processed all of the packets, the memory was restored to the memory pool as expected. The SB memory-in-use telemetry was zero because there were no software bus packets in transit. At this point any attempt to send a new-sized packet on the software bus was be rejected. The ES memory pool stated that the ". . . Request won't fit in remaining memory" even though there was currently no memory in use.

The simplest way to prevent this behavior is to ensure that there is margin when sizing the SB memory pool. To check the margin, monitor the "Peak Memory in Use" vs. the configuration parameter CFE_SB_BUF_MEMORY_BYTES which indicates the amount allocated.

Next: Frequently Asked Questions about Software Bus
Prev: Quality of Service
Up To: Operation of the SB Software

8.11.34 Frequently Asked Questions about Software Bus

(Q) How is the memory pool handle (sent in SB housekeeping telemetry) intended to be used?

The memory pool handle is used to analyze the SB memory pool statistics. The cFE ES command (CFE_ES_SEND_MEM_POOL_STATS_CC) to dump the memory pool statistics takes the pool handle as a parameter. These statistics tell how the SB memory pool is configured and gives details on margin. An improperly configured SB memory pool may inhibit communication. This may occur if there is not enough margin to create a block of the size needed for a transfer. Refer to the ES memory pool users guide for more details. Memory Pool Service

(Q) When sending a message, what message header fields are critical for routing the message?

To route the message properly, the software bus uses only the Message ID and packet length fields from the header of the message. If the packet length field is incorrect, then the buffer allocation for the message will also be incorrect. This may appear to the receiver as a truncated message or a message with unknown data added to the end of the message.

(Q) How many copies of the message are performed in a typical message delivery?

There is a single copy of the message performed during a typical delivery. During the CFE_SB_SendMsg API, the software bus copies the message from the callers memory space to the software bus memory space. The CFE_SB_RcvMsg API gives the user a pointer to the message in the software bus memory space. This is equivalent to the copy mode send and pointer mode receive in the heritage software bus used on WMAP, ST5, SDO etc.

(Q) When does the software bus free the message buffer during a typical message delivery process? Or how long is the message, and the pointer to the message in the CFE_SB_RcvMsg valid?

After receiving a message by calling CFE_SB_RcvMsg, the message received stays in the software bus memory until the next call to CFE_SB_RcvMsg with the same Pipe Id. This means that the message pointer given by the software bus to the caller of CFE_SB_RcvMsg is valid until the next call to CFE_SB_RcvMsg with the same pipe id. If the caller needs the message longer than the next call to CFE_SB_RcvMsg, the caller must copy the message to its memory space.

(Q) The first parameter in the CFE_SB_RcvMsg API is a pointer to a pointer which can get confusing. How can I be sure that the correct address is given for this parameter.

Typically a caller declares a ptr of type CFE_SB_Msg_t (i.e. CFE_SB_Msg_t *Ptr) then gives the address of that pointer (&Ptr) as this parameter. After a successful call to CFE_SB_RcvMsg, Ptr will point to the first byte of the software bus message header. This should be used as a read-only pointer. In systems with an MMU, writes to this pointer may cause a memory protection fault.

(Q) Why am I not seeing expected Message Limit error events or Pipe Overflow events?

It is possible the events are being filtered by cFE Event Services. The filtering for this event may be specified in the platform configuration file or it may have been commanded after the system initializes.
There is a corresponding counter for each of these conditions. First verify that the condition is happening by viewing the counter in SB HK telemetry. If the condition is happening, you can view the SB filter information through the EVS App Data Main page by clicking the 'go to' button for SB. The event Id for these events can be learned through a previous event or from the cfe_sb_events.h file.

(Q) Why does the SB provide event filtering through the platform configuration file?

To give the user the ability to filter events before an EVS command can be sent. During system initialization, there are many conditions occurring that can cause a flood of SB events such as No Subscribers, Pipe Overflow and MsgId to Pipe errors. This gives the user a way to limit these events.

(Q) Why does SB have so many debug event messages?

The SB debug messages are positive acknowledgments that an action (like receiving a cmd, creating a pipe or subscribing to a message) has occurred. They are intended to help isolate system problems. For instance, if an expected response to a command is not happening, it may be possible to repeat the scenario with the debug event turned on to verify that the command was successfully received.

(Q) How is the QOS parameter in the CFE_SB_SubscribeEx used by the software bus?

The QOS parameter is currently unused by the software bus. It is a placeholder to be used with the future software bus capability of inter-processor communication. Setting the QOS value to the SB defined CFE_SB_Default_↩ Qos (QOS.Priority=0,QOS.Reliability=0) will ensure seamless integration when the software bus is expanded to support inter-processor communication.

Prev: Operation of the SB Software
Up To: cFE Software Bus Overview

8.12 cFE Software Bus Commands

The following is a list of commands that are processed by the cFE Software Bus Task.

Global CFE_SB_DISABLE_ROUTE_CC
Disable Software Bus Route

Global CFE_SB_DISABLE_SUB_REPORTING_CC
Disable Subscription Reporting Command

Global CFE_SB_ENABLE_ROUTE_CC
Enable Software Bus Route

Global CFE_SB_ENABLE_SUB_REPORTING_CC
Enable Subscription Reporting Command

Global CFE_SB_NOOP_CC
Software Bus No-Op

Global CFE_SB_RESET_COUNTERS_CC
Software Bus Reset Counters

Global CFE_SB_SEND_MAP_INFO_CC
Write Map Info to a File

Global CFE_SB_SEND_PIPE_INFO_CC
Write Pipe Info to a File

Global CFE_SB_SEND_PREV_SUBS_CC
Send Previous Subscriptions Command

Global CFE_SB_SEND_ROUTING_INFO_CC
Write Software Bus Routing Info to a File

Global CFE_SB_SEND_SB_STATS_CC
Send Software Bus Statistics

8.13 cFE Software Bus Telemetry

The following are telemetry packets generated by the cFE Software Bus Task.

Class CFE_SB_AllSubscriptionsTlm_Payload_t
SB Previous Subscriptions Packet

Class CFE_SB_HousekeepingTlm_Payload_t
Software Bus task housekeeping Packet

Class CFE_SB_SingleSubscriptionTlm_Payload_t
SB Subscription Report Packet

Class CFE_SB_StatsTlm_Payload_t
SB Statistics Telemetry Packet

8.14 cFE Software Bus Configuration Parameters

The following are configuration parameters used to configure the cFE Software Bus either for each platform or for a mission as a whole.

Global CFE_MISSION_SB_MAX_PIPES
Maximum Number of pipes that SB command/telemetry messages may hold

Global CFE_MISSION_SB_MAX_SB_MSG_SIZE
Maximum SB Message Size

Global CFE_PLATFORM_ENDIAN
Platform Endian Indicator

Global CFE_PLATFORM_SB_BUF_MEMORY_BYTES
Size of the SB buffer memory pool

Global CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
Default Message Map Filename

Global CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
Default Subscription Message Limit

Global CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
Default Pipe Information Filename

Global CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER
Define Default Sender Information Storage Mode

Global CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME
Default Routing Information Filename

Global CFE_PLATFORM_SB_FILTERED_EVENT1
SB Event Filtering

Global CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
Highest Valid Message Id

Global CFE_PLATFORM_SB_MAX_DEST_PER_PKT

Maximum Number of unique local destinations a single MsgId can have

Global CFE_PLATFORM_SB_MAX_MSG_IDS

Maximum Number of Unique Message IDs SB Routing Table can hold

Global CFE_PLATFORM_SB_MAX_PIPE_DEPTH

Maximum depth allowed when creating an SB pipe

Global CFE_PLATFORM_SB_MAX_PIPES

Maximum Number of Unique Pipes SB Routing Table can hold

8.15 cFE Table Services Overview

Applications typically organize sets of their parameters into logical units called tables. These are typically constant parameters that can change the behavior of a flight software algorithm and are only intended to be modified by operations personnel. Examples of this would be attitude control gains, sensor scalefactors, telemetry filter settings, etc. None of the cFE core applications (EVS, SB, ES, TIME or TBL) use tables.

Table Services (TBL) provides a centralized control of flight software tables. Operations personnel would interact with TBL in order to dump the contents of current tables, load new table images, verify the contents of a table image and manage Critical tables.

For additional detail on Tables and how to manage them, see the following sections:

- How To Remove cFE Table Services
- Managing Tables
- cFE Table Types and Table Options
  - Single Buffered Tables
  - Double Buffered Tables
  - Critical Tables
- Table Registry
- Table Services Telemetry
- Effects of Processor Reset on Tables
- Frequently Asked Questions about Table Services

8.15.1 How To Remove cFE Table Services

It is possible to build the CFE without including Table Services. This is only applicable if the mission does not intend to use any CFS applications that require CFE type table services, or if the mission intends to provide custom table services. If CFE Table Services are removed, the CFE makefile will no longer try to make the Table Services application and the link makefile will no longer include the Table Services object module in the CFE-CORE. Even if excluded from the build, the Table Services source and header files will remain in the CFE source tree.

The variable EXCLUDE_CFE_TBL in the setvars.sh file controls whether, or not, CFE Table Services application is included in the CFE-CORE. The value of EXCLUDE_CFE_TBL must be set equal to TRUE to cause Table Services to be excluded from the CFE-CORE. Any definition of EXCLUDE_CFE_TBL that does not set the value equal to TRUE (or no definition at all) will result in the inclusion of cFE Table Services. The default setvars.sh file contains the line "# EXCLUDE_CFE_TBL=TRUE", but note that the "#" symbol marks this line as a comment. Remove the "#" symbol to enable the definition that excludes CFE Table Services.

Removing Table Services reduces the size of the CFE-CORE load file and also reduces the amount of RAM memory required to load the cFE. Each development environment will have unique savings. The numbers from a test performed using a MCP-750 platform with a GCC compiler are provided for reference:

```
Size of cFE binary load file with Table Services: 830,969
Size of cFE binary load file w/o Table services:  721,466

Amount of available RAM after loading cFE with Table Services: 76,513,488
Amount of available RAM after loading cFE w/o Table Services:  77,151,984
```

Next: Managing Tables
Up To: cFE Table Services Overview

8.15.2 Managing Tables

In order to effectively manage tables, an operator needs to understand how cFE Applications manage tables from their end. There are a number of methods that cFE Applications typically use to manage their tables. Each method is appropriate based upon the nature of the contents of the table.

cFE Applications are required to periodically check to see if their table is to be validated, updated (or in the case of dump-only tables, dumped). Most Applications perform this periodic management at the same time as housekeeping requests are processed. This table management is performed by the cFE Application that "owns" a table (ie - the cFE Application that registered the table with cFE Table Services). It is possible for cFE Applications to "share" a table with other cFE Applications. An Application that shares a table does not typically perform any of the management duties associated with that table.

A table can have one of two different types and a number of different options. These are discussed further in later sections. An operator should understand the chosen type and selected options for a particular table before attempting to modify a table's contents.

To understand the methods of maintaining a table, it is important that the terminology be clear. A table has two images: "Active" and "Inactive". The Active table is the one that a cFE Application is currently accessing when it executes. The Inactive table is a copy of the Active table that an operator (or on-board process such as a stored command processor) can manipulate and change to have a newly desired set of data.

To create an Inactive table image on board, the operator would be required to perform a "Load" to the table. Loads are table images stored in on-board files. The Load can contain either a complete table image or just a part of a table image. If the Load contains just a portion, the Inactive image is first initialized with the contents of the Active image and then

the portion identified in the Load file is written on top of the Active image. After the initial Load, an operator can continue to manipulate the Inactive table image with additional partial table load images. This allows the operator to reconfigure the contents of multiple portions of the table before deciding to "Validate" and/or "Activate" it.

Some cFE Applications provide special functions that will examine a table image to determine if the contents are logically sound. This function is referred to as the "Validation Function." When a cFE Application assigns a Validation Function to a table during the table registration process, it is then requiring that a Validation be performed before the table can be Activated. When an operator requests a Validation of a table image, they are sending a request to the owning Application to execute the associated Validation Function on that image. The results of this function are then reported in telemetry. If the Validation is successful, the operator is free to perform a table Activation. If the Validation fails, the operator would be required to make additional changes to the Inactive table image and attempt another Validation before commanding an Activation.

To change an Inactive table image into the Active table image, an operator must Activate a table. When an operator sends the table Activation command, they are notifying the table's owning Application that a new table image is available. It is then up to the Application to determine when is the best time to perform the "Update" of the table. When an Application performs an Update, the contents of the Inactive table image become the Active table image.

Next: cFE Table Types and Table Options
Prev: How To Remove cFE Table Services
Up To: cFE Table Services Overview

8.15.3 cFE Table Types and Table Options

A cFE Application Developer has several choices when creating a cFE Application. There are two basic types of tables: single buffered and double buffered. In addition to these two basic types there are a small variety of options possible with each table. These options control special characteristics of the table such as whether it is dump-only, critical or whether it has an application defined location in memory.

Each choice has its advantages and disadvantages. The developer chooses the appropriate type based upon the requirements of the application. Anyone operating a particular cFE Application must understand the nature of the type and options selected for a particular table before they can successfully understand how to perform updates, validations, etc.

For more information on the different types of tables available, see the following sections:

- Table Types
  - Single Buffered Tables
  - Double Buffered Tables
- Table Options
  - Tables with Validation Functions
  - Critical Tables

- User Defined Address Tables
- Dump Only Tables

Next: Single Buffered Tables
Prev: Managing Tables
Up To: cFE Table Services Overview

8.15.4 Single Buffered Tables

The default table type for a cFE Application to use is a single buffered table. The principle advantage of a single buffered table is that it can share one of several shared table buffers for uploaded and pending table images. Since many cFE Applications have relatively small tables that are not changed at time critical moments or are not changed very often during a mission, single buffered tables represent the most memory resource efficient method of being managed.

The number of single buffered tables that can have inactive table images being manipulated at one time is specified by a TBL Services configuration parameter (CFE_TBL_MAX_SIMULTANEOUS_LOADS) found in the cfe_platform_cfg.h file associated with the processor in question. This parameter identifies the number of shared table buffers that are available.

Since inactive single buffered table images share a common resource, it may not be prudent for an operator to load an image and then delay on the image's activation for an extended period of time.

Single buffered tables are allowed to be critical (see Critical Tables), dump-only (see Dump Only Tables) and/or have a user-defined address (see User Defined Address Tables).

Next: Double Buffered Tables
Up To: cFE Table Types and Table Options

8.15.5 Double Buffered Tables

Under certain conditions, a cFE Application Developer may choose to use a double buffered table type within their application. Double buffered tables retain a dedicated inactive image of the table data. With a dedicated inactive table image available, double buffered tables are then capable of efficiently swapping table contents and/or delaying the activation of a table's contents for an indeterminate amount of time.

Some cFE Applications prefer to delay the Activation of a table until a specified time (e.g. - a Spacecraft Ephemeris). These tables are typically defined as double buffered tables so that the Inactive image can be left sitting untouched for an extended period of time without interfering with shared resources for other tables. Then the Application can perform the Update when the time is right.

Applications which have unusually large tables may decide to conserve memory resources by making them double buffered. This is because the shared buffers used by single buffered tables must be sized to match the largest table. If there is one table that is unusually large, there is little reason to allocate up to CFE_TBL_MAX_SIMULTANEOUS_L↩OADS number of buffers that size. A double buffered table will only allocate ONE extra buffer of that size.

Performance minded Applications that are required to perform processing with tight timing deadlines may choose to use double buffered tables because the Update for a double buffered table is deterministic and quick.

Next: Tables with Validation Functions
Prev: Single Buffered Tables
Up To: cFE Table Types and Table Options

8.15.6 Tables with Validation Functions

Applications that associate Validation Functions with their tables when the tables are registered are effectively requiring that the contents of a table be logically Validated before it is Activated. The cFE will refuse to let a table with an associated Validation Function be Activated until a successful Validation on the Inactive table image has occurred.

Tables that are NOT assigned a Validation Function are assumed to be valid regardless of the contents of the table image. These tables do not require a Validation Command prior to Activation.

Next: Critical Tables
Prev: Double Buffered Tables
Up To: cFE Table Types and Table Options

8.15.7 Critical Tables

Applications that must be able to recover quickly from a Processor Reset may select the "Critical" table option when registering their table. Table Services automatically creates a Critical Data Store for the table and ensures that the contents of the Critical Data Store are updated whenever a Table Activation occurs.

If a Processor Reset happens, when the Application attempts to Register the table again, Table Services automatically locates the associated Critical Data Store and initializes the Table with the saved contents.

Next: User Defined Address Tables
Prev: Tables with Validation Functions
Up To: cFE Table Types and Table Options

8.15.8 User Defined Address Tables

In order to provide a mechanism for Flight Software Maintenance teams to quickly create a table image for dumping contents of memory that isn't normally loaded by the ground, there is an option to create User-Defined Address tables. These tables, when they are first registered, provide a memory address where the Active image of the table is to be maintained. Normally, the address is specified by Table Services from its memory pool.

By specifying the address, the Flight Software Maintenance team can create a Dump-Only table that contains the contents of a data structure that is not normally accessible via telemetry or table dumps. Then, on command, the Flight Software Maintenance team can periodically dump the data structure's contents to an on-board file(s) that can then be transferred to the ground for later analysis.

Next: Dump Only Tables
Prev: Critical Tables
Up To: cFE Table Types and Table Options

8.15.9 Dump Only Tables

On occasion, cFE Applications require a segment of memory in which the Application writes data. The typical cFE Table is not normally modified directly by an Application but only via Load and Activate commands from either the Ground or Stored Command Processor. However, for those situations where an Application wishes to modify the contents of a data structure and the Application is limited in its telemtry bandwidth so that the modified data cannot be telemetered, the Application can create a Dump-Only table.

Dump-Only tables are not allowed to be modified via the Load/Validate/Activate process most other tables are. They are only supposed to be modified by onboard Applications. The Operator can still command a Dump which will be processed by the table's owning Application when it manages its tables. By letting the Application perform the dump, the Operator can feel confident that the table contents are a complete snapshot in time and not corrupted by taking a snapshot while the Application was in the process of modifying its contents.

Next: Table Registry
Prev: User Defined Address Tables
Up To: cFE Table Types and Table Options

8.15.10 Single Buffered Tables

The default table type for a cFE Application to use is a single buffered table. The principle advantage of a single buffered table is that it can share one of several shared table buffers for uploaded and pending table images. Since many cFE Applications have relatively small tables that are not changed at time critical moments or are not changed very often during a mission, single buffered tables represent the most memory resource efficient method of being managed.

The number of single buffered tables that can have inactive table images being manipulated at one time is specified by a TBL Services configuration parameter (CFE_TBL_MAX_SIMULTANEOUS_LOADS) found in the cfe_platform_cfg.h file associated with the processor in question. This parameter identifies the number of shared table buffers that are available.

Since inactive single buffered table images share a common resource, it may not be prudent for an operator to load an image and then delay on the image's activation for an extended period of time.

Single buffered tables are allowed to be critical (see Critical Tables), dump-only (see Dump Only Tables) and/or have a user-defined address (see User Defined Address Tables).

Next: Double Buffered Tables
Up To: cFE Table Types and Table Options

8.15.11 Double Buffered Tables

Under certain conditions, a cFE Application Developer may choose to use a double buffered table type within their application. Double buffered tables retain a dedicated inactive image of the table data. With a dedicated inactive table image available, double buffered tables are then capable of efficiently swapping table contents and/or delaying the activation of a table's contents for an indeterminate amount of time.

Some cFE Applications prefer to delay the Activation of a table until a specified time (e.g. - a Spacecraft Ephemeris). These tables are typically defined as double buffered tables so that the Inactive image can be left sitting untouched for an extended period of time without interfering with shared resources for other tables. Then the Application can perform the Update when the time is right.

Applications which have unusually large tables may decide to conserve memory resources by making them double buffered. This is because the shared buffers used by single buffered tables must be sized to match the largest table. If there is one table that is unusually large, there is little reason to allocate up to CFE_TBL_MAX_SIMULTANEOUS_L↩OADS number of buffers that size. A double buffered table will only allocate ONE extra buffer of that size.

Performance minded Applications that are required to perform processing with tight timing deadlines may choose to use double buffered tables because the Update for a double buffered table is deterministic and quick.

Next: Tables with Validation Functions
Prev: Single Buffered Tables
Up To: cFE Table Types and Table Options

8.15.12 Critical Tables

Applications that must be able to recover quickly from a Processor Reset may select the "Critical" table option when registering their table. Table Services automatically creates a Critical Data Store for the table and ensures that the contents of the Critical Data Store are updated whenever a Table Activation occurs.

If a Processor Reset happens, when the Application attempts to Register the table again, Table Services automatically locates the associated Critical Data Store and initializes the Table with the saved contents.

Next: User Defined Address Tables
Prev: Tables with Validation Functions
Up To: cFE Table Types and Table Options

8.15.13 Table Registry

When Applications register tables, Table Services retains pertinent information on the table in the Table Registry. The following information (along with other information that is less important for an operator) is kept for each table:

- The Application ID of the Application that Registered the table
- The full name of the table
- The size, in bytes, of the table
- Pointers to the start addresses of the Table's image buffers, Active and Inactive (if appropriate)
- A pointer to the start address of a Validation Function
- A flag indicating whether a table image has been loaded into an Inactive buffer
- A flag indicating whether the table is Critical and its associated CDS Handle if it is
- A flag indicating whether the table has ever been loaded (initialized)
- A flag indicating whether the table is Dump Only
- A flag indicating whether the table has an Update Pending
- A flag indicating whether the table is double buffered or not
- The System Time when the Table was last Updated

- The filename of the last file loaded into the table
- The File Creation Time for the last file used to load the contents of the table

This information can be obtained by either sending the Dump Registry command which will put all of the information from the Table Registry into an onboard file for later downlink or the operator can send a command to Telemeter the Registry Entry for a single table. This will cause the pertinent registry entry for a single table to be sent via a telemetry packet.

The API function CFE_TBL_Register() returns either CFE_SUCCESS or CFE_TBL_INFO_RECOVERED_TBL to indicate that the table was successfully registered. The difference is whether the table data was recovered from CDS as part of the registration. There are several error return values that describe why the function failed to register the table but nothing related to why the restoration from CDS might have failed. There is, however, a message written to the System Error Log by Table Services that can be dumped by the ground to get this information. Note that failure to restore a table from CDS is not an expected error and requires some sort of data corruption to occur.

Next: Table Services Telemetry
Prev: cFE Table Types and Table Options
Up To: cFE Table Services Overview

8.15.14 Table Services Telemetry

Table Services produces two different telemetry packets. The first packet, referred to as the Table Services Housekeeping Packet, is routinely produced by Table Services upon receipt of the Housekeeping Request message that is typically sent to all Applications by an on board scheduler. The contents and format of this packet are described in detail at CFE_TBL_HkPacket_t.

Next: Effects of Processor Reset on Tables
Prev: Table Registry
Up To: cFE Table Services Overview

8.15.15 Effects of Processor Reset on Tables

When a processor resets, the Table Registry is re-initialized. All Applications must, therefore, re-register and re-initialize their tables. The one exception, however, is if the Application has previously tagged a table as "Critical" during Table Registration, then Table Services will attempt to locate a table image for that table stored in the Critical Data Store. Table Services also attempts to locate the Critical Table Registry which is also maintained in the Critical Data Store.

If Table Services is able to find a valid table image for a Critical table in the Critical Data Store, the contents of the table are automatically loaded into the table and the Application is notified that the table does not require additional initialization.

Next: Frequently Asked Questions about Table Services
Prev: Table Services Telemetry
Up To: cFE Table Services Overview

8.15.16 Frequently Asked Questions about Table Services

(Q) Is it an error to load a table image that is smaller than the registered size?

Table images that are smaller than the declared size of a table fall into one of two categories.
If the starting offset of the table image (as specified in the Table Image secondary file header) is not equal to zero, then the table image is considered to be a "partial" table load. Partial loads are valid as long as a table has been previously loaded with a non-"partial" table image.
If the starting offset of the table image is zero and the size is less than the declared size of the table, the image is considered "short" but valid. This feature allows application developers to use variable length tables.

(Q) I tried to validate a table and received the following event message that said the event failed:

```
"MyApp validation failed for Inactive 'MyApp.MyTable', Status=0x####"
```

What happened?

The event message indicates the application who owns the table has discovered a problem with the contents of the image. The code number following the 'Status' keyword is defined by the Application. The documentation for the specified Application should be referred to in order to identify the exact nature of the problem.

(Q) What commands do I use to load a table with a new image?

There are a number of steps required to load a table.

1. The operator needs to create a cFE Table Services compatible table image file with the desired data contained in it. This can be accomplished by creating a 'C' source file, compiling it with the appropriate cross compiler for the onboard platform and then running the `elf2cfetbl` utility on the resultant object file.
2. The file needs to be loaded into the onboard processor's filesystem using whichever file transfer protocol is used for that mission.
3. The Load Command is sent next to tell Table Services to load the table image file into the Inactive Table Image Buffer for the table identified in the file.
4. The Validate Command is then sent to validate the contents of the inactive table image. This will ensure the file was not corrupted or improperly defined. The results of the validation are reported in Table Services Housekeeping Telemetry. If a table does not have a validation function associated with it, the operator may wish to compare the computed CRC to verify the table contents match what was intended.
5. Upon successful validation, the operator then sends the Activate Command. The application owning the table should, within a reasonable amount of time, perform a table update and send an event message.

(Q) What causes cFE Table Services to generate the following sys log message:
CFE_TBL:GetAddressInternal-App(%d) attempt to access unowned Tbl Handle=%d

When an application sharing its table(s) with one or more applications is reloaded, the reloaded application's table handle(s) are released. cFE Table Services sees that the table(s) are shared and keeps a 'shadow' version of the table in the Table Services registry. The registry will show the released, shared tables with no name. When the applications sharing the table attempt to access the table via the 'old', released handle, Table Services will return an error code to the applications and generate the sys log message. The applications may then unregister the 'old' handle(s) in order to remove the released, shared table(s) from the Table Services registry and share the newly loaded application table(s).

(Q) When does the Table Services Abort Table Load command need to be issued?

The Abort command should be used whenever a table image has been loaded but the application has not yet activated it and the operator no longer wants the table to be loaded.
The purpose of the Abort command is to free a previously allocated table buffer. It should be noted, however, that multiple table loads to the SAME table without an intervening activation or abort, will simply OVERWRITE the previous table load using the SAME buffer.
Therefore, the most likely scenarios that would lead to a needed abort are as follows:

1. Operator loads a table and realizes immediately that the load is not wanted.
2. Operator loads a table and performs a validation on it. Regardless of whether the table passes or fails the validation, if the operator no longer wants to activate the table, the abort command should be issued.

   It should be noted that a table image that fails activation is retained in the inactive buffer for diagnosis, if necessary. It is NOT released until it is aborted or overwritten and successfully validated and activated.
3. A table image was loaded; the image was successfully validated; the command for activation was sent; but the application fails to perform the activation.

   The Abort command will free the table buffer and clear the activation request.

   This situation can occur when either the application is improperly designed and fails to adequately manage its tables (sometimes seen in the lab during development) or the application is "hung" and not performing as it should.

Prev: Effects of Processor Reset on Tables
Up To: cFE Table Services Overview

8.16 cFE Table Services Commands

The following is a list of commands that are processed by the cFE Table Services Task.

Global CFE_TBL_ABORT_LOAD_CC
Abort Table Load

Global CFE_TBL_ACTIVATE_CC
Activate Table

Global CFE_TBL_DELETE_CDS_CC
Delete Critical Table from Critical Data Store

Global CFE_TBL_DUMP_CC
Dump Table

Global CFE_TBL_DUMP_REGISTRY_CC
Dump Table Registry

Global CFE_TBL_LOAD_CC
Load Table

Global CFE_TBL_NOOP_CC
Table No-Op

Global CFE_TBL_RESET_COUNTERS_CC
Table Reset Counters

Global CFE_TBL_SEND_REGISTRY_CC
Telemeter One Table Registry Entry

Global CFE_TBL_VALIDATE_CC
Validate Table

8.17 cFE Table Services Telemetry

The following are telemetry packets generated by the cFE Table Services Task.

Class CFE_TBL_HousekeepingTlm_Payload_t
Table Services Housekeeping Packet

Class CFE_TBL_TblRegPacket_Payload_t
Table Registry Info Packet

8.18 cFE Table Services Configuration Parameters

The following are configuration parameters used to configure the cFE Table Services either for each platform or for a mission as a whole.

Global CFE_MISSION_TBL_MAX_FULL_NAME_LEN
Maximum Length of Full Table Name in messages

Global CFE_MISSION_TBL_MAX_NAME_LENGTH
Maximum Table Name Length

Global CFE_PLATFORM_TBL_BUF_MEMORY_BYTES
Size of Table Services Table Memory Pool

Global CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE
Default Filename for a Table Registry Dump

Global CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES
Maximum Number of Critical Tables that can be Registered

Global CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE
Maximum Size Allowed for a Double Buffered Table

Global CFE_PLATFORM_TBL_MAX_NUM_HANDLES
Maximum Number of Table Handles

Global CFE_PLATFORM_TBL_MAX_NUM_TABLES
Maximum Number of Tables Allowed to be Registered

Global CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS
Maximum Number of Simultaneous Table Validations

Global CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS
Maximum Number of Simultaneous Loads to Support

Global CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE
Maximum Size Allowed for a Single Buffered Table

Global CFE_PLATFORM_TBL_VALID_PRID_1
Processor ID values used for table load validation

Global CFE_PLATFORM_TBL_VALID_PRID_COUNT
Number of Processor ID's specified for validation

Global CFE_PLATFORM_TBL_VALID_SCID_1
Spacecraft ID values used for table load validation

Global CFE_PLATFORM_TBL_VALID_SCID_COUNT
Number of Spacecraft ID's specified for validation

8.19 cFE Time Services Overview

The cFE Time Service (TIME) is one of the cFE core services. TIME provides time correlation, distribution and synchronization services. TIME exists in two varieties: a Time Server responsible for maintaining the master time reference for all remote systems, and a Time Client responsible for synchronizing to that master time reference.

Since TIME is a generic implementation aimed to meet the needs of a variety of mission configurations, there are numerous configuration parameters, which dictate the behavior of TIME (see cfe_mission_cfg.h and cfe_platform_cfg.h for the specific mission configuration).

With the exception of those sections specific to Time Clients and Servers, this document assumes the most common physical environment - one instantiation of cFE installed on a single processor. Therefore, TIME represents cFE Time Services configured as a Time Server.

For additional detail on Time Services and how to manage it, see the following sections:

- Time Components
- Time Structure
- Time Formats
- Time Configuration
  - Time Format Selection
  - Enabling Fake Tone Signal
  - Selecting Tone and Data Ordering
  - Specifying Tone and Data Window
  - Specifying Time Server/Client
  - Specifying Time Tone Byte Order
  - Virtual MET
  - Specifying Time Source
  - Specifying Time Signal
- Time Services Paradigm(s)
- Flywheeling

8.19.1 Time Components

Time knowledge is stored in several pieces, so that the time information can more easily be manipulated and utilized. These components include:

The **Ground Epoch** is an arbitrary date and time that establishes the zero point for spacecraft time calculations. The selection of the epoch is mission specific, although in the past, it was common to select the same epoch as defined for the Operating System used by the computers hosting the ground system software. Recent mission epoch selections have also included using zero seconds after midnight, Jan 1, 2001.

Spacecraft Time is the number of seconds (and fraction of a second) since the ground epoch. Spacecraft time is the sum of **Mission Elapsed Time** (MET) and the **Spacecraft Time Correlation Factor** (STCF). By definition, MET is a measure of time since launch or separation. However, for most missions the MET actually represents the amount of time since powering on the hardware containing the MET timer. The STCF correlates the MET to the ground epoch.

The **Tone** is the signal that MET seconds have incremented. In most hardware configurations, the tone is synonymous with the **1 PPS** signal. The tone signal may be generated by a local hardware timer, or by an external event (GPS receiver, spacewire time tick, 1553 bus signal, etc). TIME may also be configured to simulate the tone for lab environments that do not have the necessary hardware to provide a tone signal. Note that MET sub-seconds will be zero at the instant of the tone.

Time at the Tone is the spacecraft time at the most recent "valid" tone.

Time since the Tone is the amount of time since the tone (usually less than one second). This value is often measured using the local processor clock. Upon detecting the tone signal, TIME stores the contents of the local processor clock to facilitate this measurement.

Thus, **Current Spacecraft Time** is the sum of "time at the tone" and "time since the tone".

Leap Seconds occur to keep clocks correlated to astronomical observations. The modern definition of a second (9,192,631,770 oscillations of a cesium-133 atom) is constant while the earth's rotation has been slow by a small fraction of a second per day. The **International Earth Rotation and Reference System Service** (IERS) maintains the count of leap seconds as a signed whole number that is subject to update twice a year. Although it is possible to have a negative leap second count if the earth rotates too fast, it is highly unlikely. The initial count of leap seconds (10) was established in January of 1972 and the first leap second was added to the initial count in June of 1972. The most recent leap seconds are announced by the International Earth Rotation Service (IERS): https://www.iers.org in IERS Bulletin C (leap second announcements). Search the IERS site for "Bulletin C" to obtain the latest issue/announcement.

Next: Time Structure
Up To: cFE Time Services Overview

8.19.2 Time Structure

The cFE implementation of the **System Time Structure** is a modified version of the CCSDS Unsegmented Time Code (CUC) which includes 4 bytes of seconds, and 4 bytes of subseconds, where a subsecond is equivalent to 1/(2^32) seconds. The system time structure is used by TIME to store current time, time at the tone, time since the tone, the MET, the STCF and command arguments for time adjustments. Note that only the 32 bits of seconds and the upper 16 bits of subseconds are used for time stamping CCSDS packets.

The system time structure is defined as follows:

```
typedef struct {
      uint32      Seconds;      /* Number of seconds */
      uint32      Subseconds;   /* Number of 2^(-32) subseconds */
} CFE_TIME_SysTime_t;
```

Next: Time Formats
Prev: Time Components
Up To: cFE Time Services Overview

8.19.3 Time Formats

International Atomic Time (TAI) is one of two time formats supported by cFE TIME. TAI is the number of seconds and sub-seconds elapsed since the ground epoch as measured with the atomic clock previously described. TAI has no reference to leap seconds and is calculated using the following equation:

```
TAI = MET + STCF
```

It should be noted that TAI is only "true" TAI when the selected ground epoch is the same as the TAI epoch (zero seconds after midnight, January 1, 1958). However, nothing precludes configuring cFE TIME to calculate time in the TAI format and setting the STCF to correlate to any other epoch definition.

Coordinated Universal Time (UTC) is the other time format supported by cFE TIME. UTC differs from TAI in the fact that UTC includes a leap seconds adjustment. TIME computes UTC using the following equation:

```
UTC = TAI - Leap Seconds.
```

The preceding UTC equation might seem to imply that TAI includes leap seconds and UTC does not - which is not the case. In fact, the UTC calculation includes a leap seconds adjustment that subtracts leap seconds from the same time components used to create TAI. Alternatively, it might be less confusing to express the UTC equation as follows:

```
UTC = MET + STCF - Leap Seconds
```

Next: Time Configuration
Prev: Time Components
Up To: cFE Time Services Overview

8.19.4 Time Configuration

All configurations of TIME require a local processor source for a 1Hz interrupt and access to a local clock with a resolution fine enough that it can be used to measure short periods of elapsed time. The local interrupt is used to wake-up TIME at a regular interval for the purpose of verifying that the tone is being received. The local clock is used to measure time since the tone and to provide coarse verification that the tone is occurring at approximately one second intervals. The presumption is that the tone is the most accurate timer in the system and, within reason, is to be trusted. Note that nothing precludes the use of the MET as the local clock, assuming the MET is both local and provides sub-second data. However, the tone must not be used as the source for the local 1Hz interrupt.

Consider the following brief description of three hypothetical hardware configurations. These sample systems may be used as reference examples to help clarify the descriptions of the various TIME configuration selections.

In the first system, there is no MET timer and therefore no tone signal. The MET is a count of the number of "fake" tones generated by TIME software. There is no validation performed regarding the quality of time data. This hardware configuration is a common lab environment using COTS equipment.

In the second system, the MET timer is a hardware register that is directly accessible by TIME. When MET seconds increment, a processor interrupt signals the tone. Upon detecting the tone, TIME can read the MET to establish the time at the tone. To verify that the tone is valid, TIME need only validate that this tone signal occurred approximately one second after the previous tone signal (as measured with the local clock).

In the third system, the MET is located on hardware connected via spacewire. When MET seconds increment, a spacewire time tick triggers a local processor interrupt to signal the tone. Shortly after announcing the tone, the hardware containing the MET also generates a spacewire data packet containing the MET value corresponding to the tone. TIME must wait until both the tone and data packet have been received before validating the tone. The tone must have occurred approximately one second after the previous tone signal and the data packet must have been received within a specified window in time following the tone.

The hardware design choice for how the tone signal is distributed is not material to TIME configuration. The software detecting the tone need only call the cFE API function announcing the arrival of the tone. This function is designed to be called from interrupt handlers.

For detail on each of the individual configuration settings for cFE Time Services, see the following sections:

- Time Format Selection
- Enabling Fake Tone Signal
- Selecting Tone and Data Ordering
- Specifying Tone and Data Window
- Specifying Time Server/Client
- Specifying Time Tone Byte Order
- Virtual MET
- Specifying Time Source
- Specifying Time Signal

Next: Time Services Paradigm(s)
Prev: Time Formats
Up To: cFE Time Services Overview

8.19.5 Time Format Selection

Time format is defined in the mission configuration header file.

This selection defines the default time format as TAI or UTC. The API functions to get time in either specific format are still enabled, but the API function to get time in the default format will follow this selection. Enable one, and **only one**, of the following time format definitions:

```
#define CFE_TIME_CFG_DEFAULT_TAI  TRUE
#define CFE_TIME_CFG_DEFAULT_UTC  FALSE
```

or

```
#define CFE_TIME_CFG_DEFAULT_TAI  FALSE
#define CFE_TIME_CFG_DEFAULT_UTC  TRUE
```

The choice of time format is a mission specific decision and is not directly affected by the hardware configuration.

See also

CFE_TIME_CFG_DEFAULT_TAI, CFE_TIME_CFG_DEFAULT_UTC

Next: Enabling Fake Tone Signal
Up To: Time Configuration

8.19.6 Enabling Fake Tone Signal

The fake tone is defined in the mission configuration header file.

If this selection is set to TRUE, TIME will generate a "fake" tone signal by calling the same API function as would be called upon detection of the "real" tone signal. Enable the fake tone only for hardware configurations that do not provide a tone signal.

```
#define CFE_TIME_CFG_FAKE_TONE  TRUE
```

Hypothetical hardware configuration number one (described above) would enable the fake tone signal.

See also

CFE_TIME_CFG_FAKE_TONE

Next: Selecting Tone and Data Ordering
Prev: Time Format Selection
Up To: Time Configuration

8.19.7 Selecting Tone and Data Ordering

Tone and data order is defined in the mission configuration header file.

This selection defines which comes first - the tone or the time at the tone data. Does the time data describe the tone that already occurred, or the tone that has not yet occurred? This decision may be driven by the hardware design but can also be arbitrary. Enable one, and only one, of the following:

```
#define CFE_TIME_AT_TONE_WAS
#define CFE_TIME_AT_TONE_WILL_BE
```

Hypothetical hardware configuration number three (described Time Configuration above) would enable "time at the tone was".

See also

CFE_TIME_AT_TONE_WAS, CFE_TIME_AT_TONE_WILL_BE

Next: Specifying Tone and Data Window
Prev: Enabling Fake Tone Signal
Up To: Time Configuration

8.19.8 Specifying Tone and Data Window

The tone and data window is defined in the mission configuration header file.

In concert with the definition of tone and data order, this selection defines the valid window in time for the second of the pair to follow the first. Both must be defined, units are micro-seconds.

```
#define CFE_TIME_MIN_ELAPSED  0
#define CFE_TIME_MAX_ELAPSED  100000
```

Hypothetical hardware configuration number three (described above) might use these values which describe a window that begins immediately after the tone and lasts for one tenth of a second.

See also

CFE_TIME_MIN_ELAPSED, CFE_TIME_MAX_ELAPSED

Next: Specifying Time Server/Client
Prev: Selecting Tone and Data Ordering
Up To: Time Configuration

8.19.9 Specifying Time Server/Client

Configure TIME as a client only when the target system has multiple processors running separate instantiations of the cFE. One instantiation must be configured as the server and the remainder configured as clients. If the target system has only one processor running the cFE, then TIME must be configured as a server.

Enable one, and only one, of the following definitions in the platform configuration header file:

```
#define CFE_TIME_CFG_SERVER   TRUE
#define CFE_TIME_CFG_CLIENT   FALSE
```

or

```
#define CFE_TIME_CFG_SERVER   FALSE
#define CFE_TIME_CFG_CLIENT   TRUE
```

See also

CFE_TIME_CFG_SERVER, CFE_TIME_CFG_CLIENT

Next: Specifying Time Tone Byte Order
Prev: Specifying Tone and Data Window
Up To: Time Configuration

8.19.10 Specifying Time Tone Byte Order

By default, the CFE time tone message is a payload of integers in platform-endian order (containing the tone's timestamp, the leap seconds, and state information.) In some configurations, it may be better to have the payload produced in big-endian order–particularly in mixed-endian environments.

In order to force the tone message to be in big-endian order, you must define the following:

```
#define CFE_PLATFORM_TIME_CFG_BIGENDIAN
```

Next: Virtual MET
Prev: Specifying Time Server/Client
Up To: Time Configuration

8.19.11 Virtual MET

This configuration option refers to whether the MET is local to this instantiation of TIME. If the MET is not local then TIME must be configured as using a virtual MET.

Therefore, all TIME clients must be configured as using a virtual MET. If the MET was local to any TIME client, then that instantiation of TIME would have to be the server.

TIME servers must be configured as using a virtual MET

Next: Specifying Time Source
Prev: Specifying Time Tone Byte Order
Up To: Time Configuration

8.19.12 Specifying Time Source

TIME configuration provides the ability to specify where the source for time data is originating - either internal or external. In hypothetical system one, the MET is internal. In system two, TIME cannot directly read the MET, therefore time data must be received from an external source.

This selection also enables a command interface to switch between internal and external input. When commanded to use internal time data, TIME will ignore the external data. However, TIME will continue to use the API function as the trigger to generate a "time at the tone" command packet regardless of the internal/external command selection.

Set the following definition to TRUE only for TIME servers using an external time data source.

```
#define CFE_TIME_CFG_SOURCE  TRUE
```

The remainder of this section pertains only to TIME servers configured to accept external time data.

When configured to accept external time data, TIME requires an additional definition for the type of external data (GPS, MET, spacecraft time, etc.). This selection will enable an API function specific to the selected data type. Regardless of how the time data is received, the receiver need only pass the data to the appropriate API function.

TIME servers using an external time data source must set one, and only one, of the following to TRUE, for example:

```
#define CFE_TIME_CFG_SRC_MET   TRUE
#define CFE_TIME_CFG_SRC_GPS   FALSE
#define CFE_TIME_CFG_SRC_TIME  FALSE
```

configuration definitions for the particular source.

If the cfe_platform_cfg.h file contains "#define CFE_TIME_CFG_SOURCE TRUE" then time is configured to allow switching between internal and external time sources (see CFE_TIME_SET_SOURCE_CC). If this configuration parameter is set to FALSE then the command to set the source will be rejected.

If this configuration parameter is set to TRUE then ONE and ONLY ONE of the following configuration parameters must also be set TRUE in order to specify the external time source, for example:

```
#define CFE_TIME_CFG_SRC_MET   TRUE
#define CFE_TIME_CFG_SRC_GPS   FALSE
#define CFE_TIME_CFG_SRC_TIME  FALSE
```

Note that Internal MET source depends on available hardware. It may be the local count of tone signals, the contents of a hardware register or an OS specific time function.

Note also that when configured to use an external time source, commands to set the time will be overwritten.

See also

CFE_TIME_CFG_SRC_MET, CFE_TIME_CFG_SRC_GPS, CFE_TIME_CFG_SRC_TIME

Next: Specifying Time Signal
Prev: Virtual MET
Up To: Time Configuration

8.19.13 Specifying Time Signal

Some hardware configurations support a primary and redundant tone signal selection. Setting the following configuration definition to TRUE will result in enabling a TIME command to select the active tone signal.

```
#define CFE_TIME_CFG_SIGNAL  TRUE
```

Note: this feature requires additional custom software to make the physical signal switch.

See also

CFE_TIME_CFG_SIGNAL

Next: Time Services Paradigm(s)
Prev: Specifying Time Source
Up To: Time Configuration

8.19.14 Time Format Selection

Time format is defined in the mission configuration header file.

This selection defines the default time format as TAI or UTC. The API functions to get time in either specific format are still enabled, but the API function to get time in the default format will follow this selection. Enable one, and **only one**, of the following time format definitions:

```
#define CFE_TIME_CFG_DEFAULT_TAI  TRUE
#define CFE_TIME_CFG_DEFAULT_UTC  FALSE
```

or

```
#define CFE_TIME_CFG_DEFAULT_TAI  FALSE
#define CFE_TIME_CFG_DEFAULT_UTC  TRUE
```

The choice of time format is a mission specific decision and is not directly affected by the hardware configuration.

See also

CFE_TIME_CFG_DEFAULT_TAI, CFE_TIME_CFG_DEFAULT_UTC

Next: Enabling Fake Tone Signal
Up To: Time Configuration

8.19.15 Enabling Fake Tone Signal

The fake tone is defined in the mission configuration header file.

If this selection is set to TRUE, TIME will generate a "fake" tone signal by calling the same API function as would be called upon detection of the "real" tone signal. Enable the fake tone only for hardware configurations that do not provide a tone signal.

```
#define CFE_TIME_CFG_FAKE_TONE  TRUE
```

Hypothetical hardware configuration number one (described above) would enable the fake tone signal.

See also

CFE_TIME_CFG_FAKE_TONE

Next: Selecting Tone and Data Ordering
Prev: Time Format Selection
Up To: Time Configuration

8.19.16 Selecting Tone and Data Ordering

Tone and data order is defined in the mission configuration header file.

This selection defines which comes first - the tone or the time at the tone data. Does the time data describe the tone that already occurred, or the tone that has not yet occurred? This decision may be driven by the hardware design but can also be arbitrary. Enable one, and only one, of the following:

```
#define CFE_TIME_AT_TONE_WAS
#define CFE_TIME_AT_TONE_WILL_BE
```

Hypothetical hardware configuration number three (described Time Configuration above) would enable "time at the tone was".

See also

CFE_TIME_AT_TONE_WAS, CFE_TIME_AT_TONE_WILL_BE

Next: Specifying Tone and Data Window
Prev: Enabling Fake Tone Signal
Up To: Time Configuration

8.19.17 Specifying Tone and Data Window

The tone and data window is defined in the mission configuration header file.

In concert with the definition of tone and data order, this selection defines the valid window in time for the second of the pair to follow the first. Both must be defined, units are micro-seconds.

```
#define CFE_TIME_MIN_ELAPSED  0
#define CFE_TIME_MAX_ELAPSED  100000
```

Hypothetical hardware configuration number three (described above) might use these values which describe a window that begins immediately after the tone and lasts for one tenth of a second.

See also

CFE_TIME_MIN_ELAPSED, CFE_TIME_MAX_ELAPSED

Next: Specifying Time Server/Client
Prev: Selecting Tone and Data Ordering
Up To: Time Configuration

8.19.18 Specifying Time Server/Client

Configure TIME as a client only when the target system has multiple processors running separate instantiations of the cFE. One instantiation must be configured as the server and the remainder configured as clients. If the target system has only one processor running the cFE, then TIME must be configured as a server.

Enable one, and only one, of the following definitions in the platform configuration header file:

```
#define CFE_TIME_CFG_SERVER   TRUE
#define CFE_TIME_CFG_CLIENT   FALSE
```

or

```
#define CFE_TIME_CFG_SERVER   FALSE
#define CFE_TIME_CFG_CLIENT   TRUE
```

See also

CFE_TIME_CFG_SERVER, CFE_TIME_CFG_CLIENT

Next: Specifying Time Tone Byte Order
Prev: Specifying Tone and Data Window
Up To: Time Configuration

8.19.19 Specifying Time Tone Byte Order

By default, the CFE time tone message is a payload of integers in platform-endian order (containing the tone's timestamp, the leap seconds, and state information.) In some configurations, it may be better to have the payload produced in big-endian order–particularly in mixed-endian environments.

In order to force the tone message to be in big-endian order, you must define the following:

```
#define CFE_PLATFORM_TIME_CFG_BIGENDIAN
```

Next: Virtual MET
Prev: Specifying Time Server/Client
Up To: Time Configuration

8.19.20 Virtual MET

This configuration option refers to whether the MET is local to this instantiation of TIME. If the MET is not local then TIME must be configured as using a virtual MET.

Therefore, all TIME clients must be configured as using a virtual MET. If the MET was local to any TIME client, then that instantiation of TIME would have to be the server.

TIME servers must be configured as using a virtual MET

Next: Specifying Time Source
Prev: Specifying Time Tone Byte Order
Up To: Time Configuration

8.19.21 Specifying Time Source

TIME configuration provides the ability to specify where the source for time data is originating - either internal or external. In hypothetical system one, the MET is internal. In system two, TIME cannot directly read the MET, therefore time data must be received from an external source.

This selection also enables a command interface to switch between internal and external input. When commanded to use internal time data, TIME will ignore the external data. However, TIME will continue to use the API function as the trigger to generate a "time at the tone" command packet regardless of the internal/external command selection.

Set the following definition to TRUE only for TIME servers using an external time data source.

```
#define CFE_TIME_CFG_SOURCE  TRUE
```

The remainder of this section pertains only to TIME servers configured to accept external time data.

When configured to accept external time data, TIME requires an additional definition for the type of external data (GPS, MET, spacecraft time, etc.). This selection will enable an API function specific to the selected data type. Regardless of how the time data is received, the receiver need only pass the data to the appropriate API function.

TIME servers using an external time data source must set one, and only one, of the following to TRUE, for example:

```
#define CFE_TIME_CFG_SRC_MET   TRUE
#define CFE_TIME_CFG_SRC_GPS   FALSE
#define CFE_TIME_CFG_SRC_TIME  FALSE
```

configuration definitions for the particular source.

If the cfe_platform_cfg.h file contains "#define CFE_TIME_CFG_SOURCE TRUE" then time is configured to allow switching between internal and external time sources (see CFE_TIME_SET_SOURCE_CC). If this configuration parameter is set to FALSE then the command to set the source will be rejected.

If this configuration parameter is set to TRUE then ONE and ONLY ONE of the following configuration parameters must also be set TRUE in order to specify the external time source, for example:

```
#define CFE_TIME_CFG_SRC_MET   TRUE
#define CFE_TIME_CFG_SRC_GPS   FALSE
#define CFE_TIME_CFG_SRC_TIME  FALSE
```

Note that Internal MET source depends on available hardware. It may be the local count of tone signals, the contents of a hardware register or an OS specific time function.

Note also that when configured to use an external time source, commands to set the time will be overwritten.

See also

CFE_TIME_CFG_SRC_MET, CFE_TIME_CFG_SRC_GPS, CFE_TIME_CFG_SRC_TIME

Next: Specifying Time Signal
Prev: Virtual MET
Up To: Time Configuration

8.19.22 Specifying Time Signal

Some hardware configurations support a primary and redundant tone signal selection. Setting the following configuration definition to TRUE will result in enabling a TIME command to select the active tone signal.

```
#define CFE_TIME_CFG_SIGNAL  TRUE
```

Note: this feature requires additional custom software to make the physical signal switch.

See also

CFE_TIME_CFG_SIGNAL

Next: Time Services Paradigm(s)
Prev: Specifying Time Source
Up To: Time Configuration

8.19.23 Time Services Paradigm(s)

In order for the cFE Time Services to work for a particular mission, the methods of obtaining time, distributing time and translating time must follow some standard paradigms used in previous missions. The following describes this expected context:

Mission dependent hardware provides the Tone. When this Tone message is received, TIME latches the local time based on the local clock. Note that in lab environments, a simulated Tone capability exists which uses an SB message. Mission dependent hardware also provides the "time at the tone" message based on the hardware latched time and the reference times stored by TIME Server. The TIME Client then updates its local reference time based on the local hardware latched time at the Tone and the provided Time-at-Tone message packet when certain checks (such as the Validity bit being set) pass.

When used in an environment that includes multiple processors, each running a separate instantiation of cFE software, the presumption is that TIME will be distributed in a client/server relationship. In this model, one processor will have TIME configured as the server and the other processors as clients. The TIME server will maintain the various time components and publish a "time at the tone" message to provide synchronized time to the TIME clients. Environments that have only a single instance of TIME must be configured as a TIME server.

In all configurations, the final step in calculating the time "right now" for any instantiation of TIME is to use a local processor clock to measure the "time since the tone".

The specific MET hardware properties will determine whether the MET value can be modified. However, the cFE design is such that there should never be a need to purposefully change or reset the MET.

Regardless of the physical hardware implementation for the MET (elapsed seconds, elapsed ticks, etc.), cFE TIME will convert the hardware MET value into a System Time Format structure for time calculations and will report the converted value in telemetry. cFE TIME will also maintain and report the STCF in a System Time Format structure.

cFE TIME has no knowledge of the current epoch; it is up to the user to keep time on the spacecraft correlated to an epoch. An exception might appear to be the epoch definition required in the cFE mission configuration definition file. However, this definition is for use only by the API functions that convert spacecraft time and file system time, and the API function that prints spacecraft time as a date and time text string. The cFE "get time" functions are independent of the ground epoch.

The mission configuration parameters, CFE_TIME_CFG_DEFAULT_TAI and CFE_TIME_CFG_DEFAULT_UTC specify the default time format. Applications are encouraged to use the CFE_TIME_GetTime API, which returns time in the format specified by this configuration parameter.

Next: Flywheeling
Prev: Time Components
Up To: cFE Time Services Overview

8.19.24 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

If the TIME server is in Flywheel mode then the TIME client is also in flywheel mode.

Next: Time State
Prev: Time Services Paradigm(s)
Up To: cFE Time Services Overview

8.19.25 Time State

Clock state is a combination of factors, most significantly whether the spacecraft time has been accurately set and whether Time Service is operating in FLYWHEEL mode. A ground command is provided to set the state to reflect when the ground has determined the spacecraft time is now correct, or that time is no longer correct. This information will be distributed to Time Clients, and in turn, to any interested sub-systems. If time has not been set then TIME services reports the state of time as invalid, regardless of whether time is flywheeling or not. Also, this command may be used to force a Time Server or Time Client into FLYWHEEL mode. Use of FLYWHEEL mode is mainly for debug purposes although, in extreme circumstances, it may be of value to force Time Service not to rely on normal time updates. Note that when commanded into FLYWHEEL mode, the Time Service will remain so until receipt of another "set state" command setting the state into a mode other than FLYWHEEL. Note also that setting the clock state to VALID or INV↩ ALID on a Time Client that is currently getting time updates from the Time Server will have very limited effect. As soon as the Time Client receives the next time update, the VALID/INVALID selection will be set to that of the Time Server. However, setting a Time Client to FLYWHEEL cannot be overridden by the Time Server since the Time Client will ignore time updates from the Time Server while in FLYWHEEL mode.

Next: Initialization
Prev: Flywheeling
Up To: cFE Time Services Overview

8.19.26 Initialization

No action is required by the ground to initialize the TIME software; however, time variables in the TIME Server must be set by command to allow correct time to propagate.

For a description of what happens during each type of reset, see below:

- Power-On Reset
- Processor Reset

Next: Power-On Reset
Prev: Time State
Up To: cFE Time Services Overview

8.19.27 Power-On Reset

TIME initializes all counters in housekeeping telemetry, sets the Validity state to Invalid, and initializes the STCF, Leap Seconds, and 1 Hz Adjustment to zero.

Next: Processor Reset
Up To: Initialization

8.19.28 Processor Reset

In the event of a processor reset, the following time values are preserved:

- MET
- STCF
- Leap Seconds
- Clock Signal Selection
- Current Time Client Delay (if applicable)

Note that since it is virtually impossible for TIME services to validate the actual data that is saved across a processor reset, a signature pattern is written to the preserved area. On a processor reset, TIME queries that signature to make sure that it matches what is expected. If the signature does not match, then TIME is initialized as if a cFE power-on reset occurred.

Next: Normal Operation
Prev: Power-On Reset
Up To: Initialization

8.19.29 Power-On Reset

TIME initializes all counters in housekeeping telemetry, sets the Validity state to Invalid, and initializes the STCF, Leap Seconds, and 1 Hz Adjustment to zero.

Next: Processor Reset
Up To: Initialization

8.19.30 Processor Reset

In the event of a processor reset, the following time values are preserved:

- MET
- STCF
- Leap Seconds
- Clock Signal Selection
- Current Time Client Delay (if applicable)

Note that since it is virtually impossible for TIME services to validate the actual data that is saved across a processor reset, a signature pattern is written to the preserved area. On a processor reset, TIME queries that signature to make sure that it matches what is expected. If the signature does not match, then TIME is initialized as if a cFE power-on reset occurred.

Next: Normal Operation
Prev: Power-On Reset
Up To: Initialization

8.19.31 Initialization

No action is required by the ground to initialize the TIME software; however, time variables in the TIME Server must be set by command to allow correct time to propagate.

For a description of what happens during each type of reset, see below:

- Power-On Reset
- Processor Reset

Next: Power-On Reset
Prev: Time State
Up To: cFE Time Services Overview

8.19.32 Power-On Reset

TIME initializes all counters in housekeeping telemetry, sets the Validity state to Invalid, and initializes the STCF, Leap Seconds, and 1 Hz Adjustment to zero.

Next: Processor Reset
Up To: Initialization

8.19.33 Processor Reset

In the event of a processor reset, the following time values are preserved:

- MET
- STCF
- Leap Seconds
- Clock Signal Selection
- Current Time Client Delay (if applicable)

Note that since it is virtually impossible for TIME services to validate the actual data that is saved across a processor reset, a signature pattern is written to the preserved area. On a processor reset, TIME queries that signature to make sure that it matches what is expected. If the signature does not match, then TIME is initialized as if a cFE power-on reset occurred.

Next: Normal Operation
Prev: Power-On Reset
Up To: Initialization

8.19.34 Power-On Reset

TIME initializes all counters in housekeeping telemetry, sets the Validity state to Invalid, and initializes the STCF, Leap Seconds, and 1 Hz Adjustment to zero.

Next: Processor Reset
Up To: Initialization

8.19.35 Processor Reset

In the event of a processor reset, the following time values are preserved:

- MET
- STCF
- Leap Seconds
- Clock Signal Selection
- Current Time Client Delay (if applicable)

Note that since it is virtually impossible for TIME services to validate the actual data that is saved across a processor reset, a signature pattern is written to the preserved area. On a processor reset, TIME queries that signature to make sure that it matches what is expected. If the signature does not match, then TIME is initialized as if a cFE power-on reset occurred.

Next: Normal Operation
Prev: Power-On Reset
Up To: Initialization

8.19.36 Normal Operation

The following sections describe the operator's resposibilities for maintaining time under nominal conditions:

- Client
- Server

Next: Client
Prev: Initialization
Up To: cFE Time Services Overview

8.19.37 Client

Under normal operation, TIME Client systems do not require any attention from the ground, however TIME clients do provide commands to set the persistent latency between the server and client. Latency can be either added or subtracted to the current TIME client time calculation to account for the latency.

Next: Server
Up To: Normal Operation

8.19.38 Server

TIME Servers require maintenance by the operations team to ensure the spacecraft is maintaining a time that can be successfully correlated to other entities. The following sections describe the commands that the operations team can use to help maintain a proper time reference:

- Setting Time
- Adjusting Time
- Setting MET

Next: Setting Time
Prev: Client
Up To: Normal Operation

8.19.38.1 Setting Time

The Time Server provides commands to set time. The new time value represents the desired offset from mission-defined time epoch and takes effect immediately upon execution of this command. Time Service will calculate a new STCF value based on the current MET and the desired new time using one of the following:

If Time Service is configured to compute current time as TAI:

```
STCF = new time - current MET
current time =  current MET + STCF
```

If Time Service is configured to compute current time as UTC:

```
STCF = ((new time) - (current MET)) + Leap Seconds
current time = ((current MET) + STCF) - Leap Seconds
```

See also

CFE_TIME_SET_TIME_CC

Next: Adjusting Time
Up To: Server

8.19.38.2 Adjusting Time

The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set by the jam command, and fine tuned using the delta adjust. TIME provides the ability to command a one time adjustment (add or subtract) to the current STCF. In addition there is a 1Hz adjustment (add or subtract) that can be made to the STCF to compensate for oscillator drift. Mission specific ground correlation should be used to assist in determining the proper values to use. The Leap Seconds should be set to the current TAI-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_SUB_ADJUST_CC, CFE_TIME_SET_STCF_CC, CFE_TIME_ADD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, CFE_TIME_SET_LEAP_SECONDS_CC

Next: Setting MET
Prev: Setting Time
Up To: Server

8.19.38.3 Setting MET

The TIME Server provides the capability to set the MET. Note that the MET (as implemented for cFE Time Service) is a logical representation and not a physical timer. Thus, setting the MET is not dependent on whether the hardware supports a MET register that can be written to. Note also that Time Service "assumes" that during normal operation, the MET is synchronized to the tone signal. Therefore, unless operating in FLYWHEEL mode, the sub-seconds portion of the MET will be set to zero at the next tone signal interrupt. The new MET takes effect immediately upon execution of this command.

See also

CFE_TIME_SET_MET_CC

Next: Frequently Asked Questions
Prev: Adjusting Time
Up To: Server

8.19.39 Client

Under normal operation, TIME Client systems do not require any attention from the ground, however TIME clients do provide commands to set the persistent latency between the server and client. Latency can be either added or subtracted to the current TIME client time calculation to account for the latency.

Next: Server
Up To: Normal Operation

8.19.40 Server

TIME Servers require maintenance by the operations team to ensure the spacecraft is maintaining a time that can be successfully correlated to other entities. The following sections describe the commands that the operations team can use to help maintain a proper time reference:

- Setting Time
- Adjusting Time
- Setting MET

Next: Setting Time
Prev: Client
Up To: Normal Operation

8.19.40.1 Setting Time

The Time Server provides commands to set time. The new time value represents the desired offset from mission-defined time epoch and takes effect immediately upon execution of this command. Time Service will calculate a new STCF value based on the current MET and the desired new time using one of the following:

If Time Service is configured to compute current time as TAI:

```
STCF = new time - current MET
current time =  current MET + STCF
```

If Time Service is configured to compute current time as UTC:

```
STCF = ((new time) - (current MET)) + Leap Seconds
current time = ((current MET) + STCF) - Leap Seconds
```

See also

CFE_TIME_SET_TIME_CC

Next: Adjusting Time
Up To: Server

8.19.40.2 Adjusting Time

The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set by the jam command, and fine tuned using the delta adjust. TIME provides the ability to command a one time adjustment (add or subtract) to the current STCF. In addition there is a 1Hz adjustment (add or subtract) that can be made to the STCF to compensate for oscillator drift. Mission specific ground correlation should be used to assist in determining the proper values to use. The Leap Seconds should be set to the current TAI-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_SUB_ADJUST_CC, CFE_TIME_SET_STCF_CC, CFE_TIME_ADD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, CFE_TIME_SET_LEAP_SECONDS_CC

Next: Setting MET
Prev: Setting Time
Up To: Server

8.19.40.3 Setting MET

The TIME Server provides the capability to set the MET. Note that the MET (as implemented for cFE Time Service) is a logical representation and not a physical timer. Thus, setting the MET is not dependent on whether the hardware supports a MET register that can be written to. Note also that Time Service "assumes" that during normal operation, the MET is synchronized to the tone signal. Therefore, unless operating in FLYWHEEL mode, the sub-seconds portion of the MET will be set to zero at the next tone signal interrupt. The new MET takes effect immediately upon execution of this command.

See also

CFE_TIME_SET_MET_CC

Next: Frequently Asked Questions
Prev: Adjusting Time
Up To: Server

8.19.41 Setting Time

The Time Server provides commands to set time. The new time value represents the desired offset from mission-defined time epoch and takes effect immediately upon execution of this command. Time Service will calculate a new STCF value based on the current MET and the desired new time using one of the following:

If Time Service is configured to compute current time as TAI:

```
STCF = new time - current MET
current time =  current MET + STCF
```

If Time Service is configured to compute current time as UTC:

```
STCF = ((new time) - (current MET)) + Leap Seconds
current time = ((current MET) + STCF) - Leap Seconds
```

See also

CFE_TIME_SET_TIME_CC

Next: Adjusting Time
Up To: Server

8.19.42 Adjusting Time

The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set by the jam command, and fine tuned using the delta adjust. TIME provides the ability to command a one time adjustment (add or subtract) to the current STCF. In addition there is a 1Hz adjustment (add or subtract) that can be made to the STCF to compensate for oscillator drift. Mission specific ground correlation should be used to assist in determining the proper values to use. The Leap Seconds should be set to the current TAI-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_SUB_ADJUST_CC, CFE_TIME_SET_STCF_CC, CFE_TIME_A↩
DD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC, CFE_TIME_SET_LEAP_SECOND↩
S_CC

Next: Setting MET
Prev: Setting Time
Up To: Server

8.19.43 Setting MET

The TIME Server provides the capability to set the MET. Note that the MET (as implemented for cFE Time Service) is a logical representation and not a physical timer. Thus, setting the MET is not dependent on whether the hardware supports a MET register that can be written to. Note also that Time Service "assumes" that during normal operation, the MET is synchronized to the tone signal. Therefore, unless operating in FLYWHEEL mode, the sub-seconds portion of the MET will be set to zero at the next tone signal interrupt. The new MET takes effect immediately upon execution of this command.

See also

CFE_TIME_SET_MET_CC

Next: Frequently Asked Questions
Prev: Adjusting Time
Up To: Server

8.19.44 Frequently Asked Questions

(Q)	

Prev: Normal Operation
Up To: cFE Time Services Overview

8.20 cFE Time Services Commands

The following is a list of commands that are processed by the cFE Time Services Task.

Global CFE_TIME_ADD_1HZ_ADJUSTMENT_CC
Add Delta to Spacecraft Time Correlation Factor each 1Hz

Global CFE_TIME_ADD_ADJUST_CC
Add Delta to Spacecraft Time Correlation Factor

Global CFE_TIME_ADD_DELAY_CC
Add Time to Tone Time Delay

Global CFE_TIME_NOOP_CC
Time No-Op

Global CFE_TIME_RESET_COUNTERS_CC
Time Reset Counters

Global CFE_TIME_SEND_DIAGNOSTIC_TLM_CC
Request TIME Diagnostic Telemetry

Global CFE_TIME_SET_LEAP_SECONDS_CC
Set Leap Seconds

Global CFE_TIME_SET_MET_CC
Set Mission Elapsed Time

Global CFE_TIME_SET_SIGNAL_CC
Set Tone Signal Source

Global CFE_TIME_SET_SOURCE_CC
Set Time Source

Global CFE_TIME_SET_STATE_CC
Set Time State

Global CFE_TIME_SET_STCF_CC
Set Spacecraft Time Correlation Factor

Global CFE_TIME_SET_TIME_CC
Set Spacecraft Time

Global CFE_TIME_SUB_1HZ_ADJUSTMENT_CC
Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

Global CFE_TIME_SUB_ADJUST_CC
Subtract Delta from Spacecraft Time Correlation Factor

Global CFE_TIME_SUB_DELAY_CC
Subtract Time from Tone Time Delay

8.21 cFE Time Services Telemetry

The following are telemetry packets generated by the cFE Time Services Task.

Class CFE_TIME_DiagnosticTlm_Payload_t
Time Services Diagnostics Packet

Class CFE_TIME_HousekeepingTlm_Payload_t
Time Services Housekeeping Packet

8.22 cFE Time Services Configuration Parameters

The following are configuration parameters used to configure the cFE Time Services either for each platform or for a mission as a whole.

Global CFE_MISSION_TIME_AT_TONE_WAS
Default Time and Tone Order

Global CFE_MISSION_TIME_CFG_DEFAULT_TAI
Default Time Format

Global CFE_MISSION_TIME_CFG_FAKE_TONE
Default Time Format

Global CFE_MISSION_TIME_DEF_MET_SECS
Default Time Values

Global CFE_MISSION_TIME_EPOCH_YEAR
Default EPOCH Values

Global CFE_MISSION_TIME_FS_FACTOR
Time File System Factor

Global CFE_MISSION_TIME_MIN_ELAPSED
Min and Max Time Elapsed

Global CFE_PLATFORM_TIME_CFG_LATCH_FLY
Define Periodic Time to Update Local Clock Tone Latch

Global CFE_PLATFORM_TIME_CFG_SERVER
Time Server or Time Client Selection

Global CFE_PLATFORM_TIME_CFG_SIGNAL
Include or Exclude the Primary/Redundant Tone Selection Cmd

Global CFE_PLATFORM_TIME_CFG_SOURCE
Include or Exclude the Internal/External Time Source Selection Cmd

Global CFE_PLATFORM_TIME_CFG_SRC_MET
Choose the External Time Source for Server only

Global CFE_PLATFORM_TIME_CFG_START_FLY
Define Time to Start Flywheel Since Last Tone

Global CFE_PLATFORM_TIME_CFG_TONE_LIMIT

Define Timing Limits From One Tone To The Next

Global CFE_PLATFORM_TIME_CFG_VIRTUAL

Time Tone In Big-Endian Order

Local MET or Virtual MET Selection for Time Servers

Global CFE_PLATFORM_TIME_MAX_DELTA_SECS

Define the Max Delta Limits for Time Servers using an Ext Time Source

Global CFE_PLATFORM_TIME_MAX_LOCAL_SECS

Define the Local Clock Rollover Value in seconds and subseconds

Global CFE_PLATFORM_TIME_START_TASK_PRIORITY

Define TIME Task Priorities

Global CFE_PLATFORM_TIME_START_TASK_STACK_SIZE

Define TIME Task Stack Sizes

8.23 cFE Event Message Cross Reference

The following cross reference maps the text associated with each cFE Event Message to its Event Message Identifier. A user can search this page for the text of the message they wish to learn more about and then click on the associated Event Message Identifier to obtain more information.

Global CFE_ES_ALL_APPS_EID

```
'App Info file written to %s, Entries=%d, FileSize=%d'
```

Global CFE_ES_BOOT_ERR_EID

```
'Invalid cFE restart type %d'
```

Global CFE_ES_BUILD_INF_EID

```
'Build s s'
```

Global CFE_ES_CC1_ERR_EID

```
'Invalid ground command code:  ID = 0x%X, CC = %d'
```

Global CFE_ES_CDS_DELETE_ERR_EID

```
'Error while deleting '%s' from CDS, See SysLog.(Err=0x%08X)'
```

Global CFE_ES_CDS_DELETE_TBL_ERR_EID

```
'CDS '%s' is a Critical Table CDS. Must be deleted via TBL Command'
```

Global CFE_ES_CDS_DELETED_INFO_EID

```
'Successfully removed '%s' from CDS'
```

Global CFE_ES_CDS_DUMP_ERR_EID

```
'Error writing CDS Registry to '%s', Status=0x%08X'
```

Global CFE_ES_CDS_NAME_ERR_EID

```
'Unable to locate '%s' in CDS Registry'
```

Global CFE_ES_CDS_OWNER_ACTIVE_EID

```
'CDS '%s' not deleted because owning app is active'
```

Global CFE_ES_CDS_REG_DUMP_INF_EID

```
'Successfully dumped CDS Registry to '%s':Size=%d,Entries=%d'
```

Global CFE_ES_CDS_REGISTER_ERR_EID

```
'%s Failed to Register CDS '%s', Status=0x%08X'
```

Global CFE_ES_CREATING_CDS_DUMP_ERR_EID

```
'Error creating CDS dump file '%s', Status=0x%08X'
```

Global CFE_ES_ERLOG1_INF_EID

```
'Cleared mode log data'
```

Global CFE_ES_ERLOG2_EID

```
'%s written:Size=%d'
```

Global CFE_ES_ERLOG2_ERR_EID

```
'Error creating file %s, stat=0x%x'
```

Global CFE_ES_ERR_SYSLOGMODE_EID

```
'Set OverWriteSysLog Command:  Invalid Mode setting = %d'
```

Global CFE_ES_ERREXIT_APP_ERR_EID

```
'Exit Application %s on Error Failed:  CleanUpApp Error 0x%08X.'
```

Global CFE_ES_ERREXIT_APP_INF_EID

```
'Exit Application %s Completed.'
```

Global CFE_ES_EXIT_APP_ERR_EID

```
'Exit Application %s Failed:  CleanUpApp Error 0x%08X.'
```

Global CFE_ES_EXIT_APP_INF_EID

```
'Exit Application %s Completed.'
```

Global CFE_ES_FILEWRITE_ERR_EID

```
'File write,byte cnt err,file %s,request=%d,actual=%d'
```

Global CFE_ES_INIT_INF_EID

```
'cFE ES Initialized'
```

Global CFE_ES_INITSTATS_INF_EID

```
'cFE Version %d.%d.%d chksm %d, OSAL Version %d.%d'
```

Global CFE_ES_INVALID_POOL_HANDLE_ERR_EID

```
'Cannot telemeter memory pool stats.  Illegal Handle (0x%08X)'
```

Global CFE_ES_LEN_ERR_EID

```
'Invalid cmd length:  ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'
```

Global CFE_ES_MID_ERR_EID

```
'Invalid command pipe message ID: 0x%X'
```

Global CFE_ES_NOOP_INF_EID

```
'No-op command'
```

Global CFE_ES_ONE_APP_EID

```
'Sent %s application data'
```

Global CFE_ES_ONE_APPID_ERR_EID

```
'Failed to send %s application data:  GetAppIDByName Failed, RC = 0x%08X'
```

Global CFE_ES_ONE_ERR_EID

```
'Failed to send %s application data, RC = %08X'
```

Global CFE_ES_OSCREATE_ERR_EID

```
'Failed to write App Info file, OS_creat returned %d'
```

Global CFE_ES_PCR_ERR1_EID

'ES_ProcControlReq: Invalid State (EXCEPTION) Application %s.'

Global CFE_ES_PCR_ERR2_EID

'ES_ProcControlReq: Unknown State (%d) Application %s.'

Global CFE_ES_PERF_DATAWRITTEN_EID

'%s written:Size=%d,EntryCount=%d'

Global CFE_ES_PERF_FILTMSKCMD_EID

'Set Performance Filter Mask command'

Global CFE_ES_PERF_FILTMSKERR_EID

'Error:Performance Filter Mask Index value greater than CFE_ES_PERF_32BIT_↩
WORDS_IN_MASK (which is a whole number derived from CFE_PLATFORM_ES_PERF_M↩
AX_IDS / 32)'

Global CFE_ES_PERF_LOG_ERR_EID

'Error creating file %s, stat=%d'

Global CFE_ES_PERF_STARTCMD_EID

'Start collecting performance data command, trigger mode = d'

Global CFE_ES_PERF_STARTCMD_ERR_EID

'Cannot start collecting performance data,perf data write in progress'

Global CFE_ES_PERF_STARTCMD_TRIG_ERR_EID

'Cannot start collecting performance data, trigger mode (d) out of range (d
to d)'

Global CFE_ES_PERF_STOPCMD_EID

'Perf Stop Cmd Rcvd,%s will write %d entries.%dmS dly every %d entries'

Global CFE_ES_PERF_STOPCMD_ERR1_EID

'Stop performance data cmd,Error creating child task RC=0x%08X'

Global CFE_ES_PERF_STOPCMD_ERR2_EID

'Stop performance data cmd ignored,perf data write in progress'

Global CFE_ES_PERF_TRIGMSKCMD_EID

'Set Performance Trigger Mask command'

Global CFE_ES_PERF_TRIGMSKERR_EID

'Error: Performance Trigger Mask Index value greater than CFE_ES_PERF_32B↩
IT_WORDS_IN_MASK (which is a whole number derived from CFE_PLATFORM_ES_PER↩
F_MAX_IDS / 32)'

Global CFE_ES_RELOAD_APP_DBG_EID

'Reload Application %s Initiated.'

Global CFE_ES_RELOAD_APP_ERR1_EID

'Failed to reload Application %s, rc = %08X'

Global CFE_ES_RELOAD_APP_ERR2_EID

'Reload Application %s, GetAppIDByName failed. RC = 0x%08X.'

Global CFE_ES_RELOAD_APP_ERR3_EID

'Reload Application %s Failed: AppCreate Error 0x%08X.'

Global CFE_ES_RELOAD_APP_ERR4_EID

'Reload Application %s Failed: CleanUpApp Error 0x%08X.'

Global CFE_ES_RELOAD_APP_INF_EID

```
'Reload Application %s Completed.'
```

Global CFE_ES_RESET_INF_EID

```
'Reset Counters command'
```

Global CFE_ES_RESET_PR_COUNT_EID

```
'Reset Processor Reset Count to Zero'
```

Global CFE_ES_RESTART_APP_DBG_EID

```
'Restart Application %s Initiated.'
```

Global CFE_ES_RESTART_APP_ERR1_EID

```
'Restart Application %s Failed, RC = 0x%08X'
```

Global CFE_ES_RESTART_APP_ERR2_EID

```
'Restart Application %s, GetAppIDByName failed.  RC = 0x%08X.'
```

Global CFE_ES_RESTART_APP_ERR3_EID

```
'Restart Application %s Failed:  AppCreate Error 0x%08X.'
```

Global CFE_ES_RESTART_APP_ERR4_EID

```
'Restart Application %s Failed:  CleanUpApp Error 0x%08X.'
```

Global CFE_ES_RESTART_APP_INF_EID

```
'Restart Application %s Completed.'
```

Global CFE_ES_RST_ACCESS_EID

```
'Error accessing ER Log,%s not written.Stat=0x%08x'
```

Global CFE_ES_SET_MAX_PR_COUNT_EID

```
'Maximum Processor Reset Count set to:  %d'
```

Global CFE_ES_SHELL_ERR_EID

```
'Failed to invoke shell command %s, rc = %08X'
```

Global CFE_ES_SHELL_INF_EID

```
'Invoked shell command %s'
```

Global CFE_ES_START_ERR_EID

```
'Failed to start %s from %s, RC = %08X'
```

Global CFE_ES_START_EXC_ACTION_ERR_EID

```
'CFE_ES_StartAppCmd:  Invalid Exception Action:  %d.'
```

Global CFE_ES_START_INF_EID

```
'Started %s from %s, AppID = %d'
```

Global CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID

```
'CFE_ES_StartAppCmd:  App Entry Point is NULL.'
```

Global CFE_ES_START_INVALID_FILENAME_ERR_EID

```
'CFE_ES_StartAppCmd:  invalid filename:  %s'
```

Global CFE_ES_START_NULL_APP_NAME_ERR_EID

```
'CFE_ES_StartAppCmd:  App Name is NULL.'
```

Global CFE_ES_START_PRIORITY_ERR_EID

```
'CFE_ES_StartAppCmd:  Priority is too large:  %d.'
```

Global CFE_ES_START_STACK_ERR_EID

```
'CFE_ES_StartAppCmd:  Stack size is less than system Minimum:  %d.'
```

Global CFE_ES_STOP_DBG_EID

```
'Stop Application %s Initiated.'
```

Global CFE_ES_STOP_ERR1_EID

```
'Stop Application %s Failed, RC = 0x%08X'
```

Global CFE_ES_STOP_ERR2_EID

```
'Stop Application %s, GetAppIDByName failed.  RC = 0x%08X.'
```

Global CFE_ES_STOP_ERR3_EID

```
'Stop Application %s Failed:  CleanUpApp Error 0x%08X.'
```

Global CFE_ES_STOP_INF_EID

```
'Stop Application %s Completed.'
```

Global CFE_ES_SYSLOG1_INF_EID

```
'Cleared Executive Services log data'
```

Global CFE_ES_SYSLOG2_EID

```
'%s written:Size=%d,Entries=%d'
```

Global CFE_ES_SYSLOG2_ERR_EID

```
'Error creating file %s, stat=0x%x'
```

Global CFE_ES_SYSLOGMODE_EID

```
'Set OverWriteSysLog Command Received with Mode setting = %d'
```

Global CFE_ES_TASKINFO_EID

```
'Task Info file written to %s, Entries=%d, FileSize=%d'
```

Global CFE_ES_TASKINFO_OSCREATE_ERR_EID

```
'Failed to write Task Info file, OS_creat returned %d'
```

Global CFE_ES_TASKINFO_WR_ERR_EID

```
'Failed to write Task Info file, Task write RC = 0x%08X, exp %d'
```

Global CFE_ES_TASKINFO_WRHDR_ERR_EID

```
'Failed to write Task Info file, WriteHdr rtnd %08X, exp %d'
```

Global CFE_ES_TASKWR_ERR_EID

```
'Failed to write App Info file, Task write RC = 0x%08X, exp %d'
```

Global CFE_ES_TLM_POOL_STATS_INFO_EID

```
'Successfully telemetered memory pool stats for 0x%08X'
```

Global CFE_ES_VERSION_INF_EID

```
'Mission s.s, s, s'
```

Global CFE_ES_WRHDR_ERR_EID

```
'Failed to write App Info file, WriteHdr rtnd %08X, exp %d'
```

Global CFE_ES_WRITE_CFE_HDR_ERR_EID

```
'Error writing cFE File Header to '%s', Status=0x%08X'
```

Global CFE_EVS_ADDFILTER_EID

```
'Add Filter Command Received with AppName = %s, EventID = 0x%08x, Mask =
0x%04x'
```

Global CFE_EVS_DELFILTER_EID

```
'Delete Filter Command Received with AppName = %s, EventID = 0x%08x'
```

Global CFE_EVS_DISAPPENTTYPE_EID

'Disable App Event Type Command Received with AppName = %s, EventType Bit Mask = 0x%02x'

Global CFE_EVS_DISAPPEVT_EID

'Disable App Events Command Received with AppName = %s'

Global CFE_EVS_DISEVTTYPE_EID

'Disable Event Type Command Received with Event Type Bit Mask = 0x%02x'

Global CFE_EVS_DISPORT_EID

'Disable Ports Command Received with Port Bit Mask = 0x%02x'

Global CFE_EVS_ENAAPPEVT_EID

'Enable App Events Command Received with AppName = %s'

Global CFE_EVS_ENAAPPEVTTYPE_EID

'Enable App Event Type Command Received with AppName = %s, EventType Bit Mask = 0x%02x'

Global CFE_EVS_ENAEVTTYPE_EID

'Enable Event Type Command Received with Event Type Bit Mask = 0x%02x'

Global CFE_EVS_ENAPORT_EID

'Enable Ports Command Received with Port Bit Mask = 0x%02x'

Global CFE_EVS_ERR_APPNOREGS_EID

'%s not registered with EVS: CC = %lu'

Global CFE_EVS_ERR_CC_EID

'Invalid command code - ID = 0x%08x, CC = %d'

Global CFE_EVS_ERR_CRDATFILE_EID

'Write App Data Command Error: OS_creat = 0x%08X, filename = %s'

Global CFE_EVS_ERR_CRLOGFILE_EID

'Write Log File Command Error: OS_creat = 0x%08X, filename = %s'

Global CFE_EVS_ERR_EVTIDNOREGS_EID

'%s Event ID %d not registered for filtering: CC = %lu'

Global CFE_EVS_ERR_ILLAPPIDRANGE_EID

'Illegal application ID %d retrieved for %s: CC = %lu'

Global CFE_EVS_ERR_ILLEGALFMTMOD_EID

'Set Event Format Mode Command: Invalid Event Format Mode = 0x%02x'

Global CFE_EVS_ERR_INVALID_BITMASK_EID

'Bit Mask = 0x%X out of range: CC = %lu'

Global CFE_EVS_ERR_LOGMODE_EID

'Set Log Mode Command Error: Log Mode = %d

Global CFE_EVS_ERR_MAXREGSFILTER_EID

'Add Filter Command: number of registered filters has reached max = %d'

Global CFE_EVS_ERR_MSGID_EID

'Invalid command packet, Message ID = 0x%08X'

Global CFE_EVS_ERR_NOAPPIDFOUND_EID

'Unable to retrieve application ID for %s: CC = %lu'

Global CFE_EVS_ERR_UNREGISTERED_EVS_APP

```
'App %s not registered with Event Services.  Unable to send event'
```

Global CFE_EVS_ERR_WRDATFILE_EID

```
'Write App Data Command Error:  OS_write = 0x%08X, filename = %s'
```

Global CFE_EVS_ERR_WRLOGFILE_EID

```
'Write Log File Command Error:  OS_write = 0x%08X, filename = %s'
```

Global CFE_EVS_EVT_FILTERED_EID

```
'Add Filter Command:AppName = %s, EventID = 0x%08x is already registered for
filtering'
```

Global CFE_EVS_FILTER_MAX_EID

```
'Max filter count reached, AppName = %s, EventID = 0x%08x:  Filter locked
until reset'
```

Global CFE_EVS_LEN_ERR_EID

```
'Invalid cmd length:  ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'
```

Global CFE_EVS_LOGMODE_EID

```
'Set Log Mode Command Error:  Log Mode = %d'
```

Global CFE_EVS_NO_LOGCLR_EID

```
'Clear Log Command:  Event Log is Disabled'
```

Global CFE_EVS_NO_LOGSET_EID

```
'Set Log Mode Command:  Event Log is Disabled'
```

Global CFE_EVS_NO_LOGWR_EID

```
'Write Log Command:  Event Log is Disabled'
```

Global CFE_EVS_NOOP_EID

```
'No-op command'
```

Global CFE_EVS_RSTALLFILTER_EID

```
'Reset All Filters Command Received with AppName = %s'
```

Global CFE_EVS_RSTCNT_EID

```
'Reset Counters Command Received'
```

Global CFE_EVS_RSTEVTCNT_EID

```
'Reset Event Counter Command Received with AppName = %s'
```

Global CFE_EVS_RSTFILTER_EID

```
'Reset Filter Command Received with AppName = %s, EventID = 0x%08x'
```

Global CFE_EVS_SETEVTFMTMOD_EID

```
'Set Event Format Mode Command Received with Mode = 0x%02x'
```

Global CFE_EVS_SETFILTERMSK_EID

```
'Set Filter Mask Command Received with AppName=%s, EventID=0x%08x, Mask=0x%04x'
```

Global CFE_EVS_STARTUP_EID

```
'cFE EVS Initialized'
```

Global CFE_EVS_WRDAT_EID

```
'Write App Data Command:  %d application data entries written to %s'
```

Global CFE_EVS_WRLOG_EID

```
'Write Log File Command:  %d event log entries written to %s'
```

Global CFE_SB_BAD_CMD_CODE_EID

```
'Invalid Cmd, Unexpected Command Code %d'
```

Global CFE_SB_BAD_MSGID_EID

```
'Invalid Cmd, Unexpected Msg Id:  0x%04x'
```

Global CFE_SB_BAD_PIPEID_EID

```
'Rcv Err:PipeId %d does not exist,app %s'
```

Global CFE_SB_CMD0_RCVD_EID

```
'No-op Cmd Rcvd'
```

Global CFE_SB_CMD1_RCVD_EID

```
'Reset Counters Cmd Rcvd'
```

Global CFE_SB_CR_PIPE_BAD_ARG_EID

```
'CreatePipeErr:Bad Input Arg:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'
```

Global CFE_SB_CR_PIPE_ERR_EID

```
'CreatePipeErr:OS_QueueCreate returned %d,app %s'
```

Global CFE_SB_CR_PIPE_NAME_TAKEN_EID

```
'CreatePipeErr:Name Taken:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'
```

Global CFE_SB_CR_PIPE_NO_FREE_EID

```
'CreatePipeErr:No Free:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'
```

Global CFE_SB_DEL_PIPE_ERR1_EID

```
'Pipe Delete Error:Bad Argument,PipedId %d,Requestor %s,Idx %d,Stat %d'
```

Global CFE_SB_DEL_PIPE_ERR2_EID

```
'Pipe Delete Error:Caller(%s) is not the owner of pipe %d'
```

Global CFE_SB_DEST_BLK_ERR_EID

```
'Subscribe Err:Request for Destination Blk failed for Msg 0x%x,Pipe %s'
```

Global CFE_SB_DSBL_RTE1_EID

```
'Disable Route Cmd:Route does not exist,Msg 0x%x,Pipe %d'
```

Global CFE_SB_DSBL_RTE2_EID

```
'Route Disabled,Msg 0x%x,Pipe %d'
```

Global CFE_SB_DSBL_RTE3_EID

```
'Disable Route Cmd:Invalid Param.Msg 0x%x,Pipe %d'
```

Global CFE_SB_DUP_SUBSCRIP_EID

```
'Duplicate Subscription,MsgId 0x%x on %s pipe,app %s'
```

Global CFE_SB_ENBL_RTE1_EID

```
'Enbl Route Cmd:Route does not exist.Msg 0x%x,Pipe %d'
```

Global CFE_SB_ENBL_RTE2_EID

```
'Enabling Route,Msg 0x%x,Pipe %d'
```

Global CFE_SB_ENBL_RTE3_EID

```
'Enbl Route Cmd:Invalid Param.Msg 0x%x,Pipe %d'
```

Global CFE_SB_FILEWRITE_ERR_EID

```
'File write,byte cnt err,file %s,request=%d,actual=%d'
```

Global CFE_SB_FULL_SUB_PKT_EID

```
'Full Sub Pkt %d Sent,Entries=%d,Stat=0x%x
'
```

Global CFE_SB_GET_BUF_ERR_EID

```
'Send Err:Request for Buffer Failed.  MsgId 0x%x,app %s,size %d'
```

Global CFE_SB_GETPIPEIDBYNAME_EID

```
'GetPipeIdByName:  ID retrieved.  Name %s,IdOut 0x%x, app %s'
```

Global CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID

```
'GetPipeIdByName Err:Name not found,Name %s,IdOut 0xx,App %s'
```

Global CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID

```
'GetPipeIdByName Err:Bad input argument,Name 0x%x,IdOut 0xx,App %s'
```

Global CFE_SB_GETPIPENAME_EID

```
'GetPipeName:  Name retrieved.  NameOut %s,Id %d, app %s'
```

Global CFE_SB_GETPIPENAME_ID_ERR_EID

```
'GetPipeName:  Id error.  NameOut %s,Id %d, app %s'
```

Global CFE_SB_GETPIPENAME_NULL_PTR_EID

```
'GetPipeName:  Null ptr error.  Id %d, app %s'
```

Global CFE_SB_GETPIPEOPTS_EID

```
'GetPipeOpts:  Options retrieved.  app %s'
```

Global CFE_SB_GETPIPEOPTS_ID_ERR_EID

```
'GetPipeOptsErr:Invalid pipe id (%d).app %s'
```

Global CFE_SB_GETPIPEOPTS_PTR_ERR_EID

```
'GetPipeOptsErr:Invalid opts ptr.app %s'
```

Global CFE_SB_GLS_INV_CALLER_EID

```
'SB GetLastSender Err:Caller(%s) is not the owner of pipe %d'
```

Global CFE_SB_INIT_EID

```
'cFE SB Initialized'
```

Global CFE_SB_LEN_ERR_EID

```
'Invalid cmd length:  ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'
```

Global CFE_SB_LSTSNDER_ERR1_EID

```
'SB GetLastSender Err:Rcvd Null Ptr,Pipe=d,App=s'
```

Global CFE_SB_LSTSNDER_ERR2_EID

```
'SB GetLastSender Err:Rcvd Invalid Pipe=d,App=s'
```

Global CFE_SB_MAX_DESTS_MET_EID

```
'Subscribe Err:Max Dests(%d)In Use For Msg 0x%x,pipe %s,app %s'
```

Global CFE_SB_MAX_MSGS_MET_EID

```
'Subscribe Err:Max Msgs(%d)In Use,MsgId 0x%x,pipe %s,app %s'
```

Global CFE_SB_MAX_PIPES_MET_EID

```
'CreatePipeErr:Max Pipes(%d)In Use.app %s'
```

Global CFE_SB_MSG_TOO_BIG_EID

```
'Send Err:Msg Too Big MsgId=0x%x,app=%s,size=%d,MaxSz=%d'
```

Global CFE_SB_MSGID_LIM_ERR_EID

```
'Send Err:Msg Limit Err MsgId 0x%x,pipe %s,sender %s'
```

Global CFE_SB_PART_SUB_PKT_EID

```
'Partial Sub Pkt %d Sent,Entries=%d,Stat=0x%x'
```

Global CFE_SB_PIPE_ADDED_EID

```
'Pipe Created:name %s,id %d,app %s'
```

Global CFE_SB_PIPE_DELETED_EID

```
'Pipe Deleted:id %d,owner %s'
```

Global CFE_SB_Q_FULL_ERR_EID

```
'Pipe Overflow,MsgId 0x%x,pipe %s,stat 0x%x,app %s'
```

Global CFE_SB_Q_RD_ERR_EID

```
'Pipe Read Err,pipe %s,app %s,stat 0x%x'
```

Global CFE_SB_Q_WR_ERR_EID

```
'Pipe Write Err,MsgId 0x%x,pipe %s,stat 0x%x,app %s'
```

Global CFE_SB_RCV_BAD_ARG_EID

```
'Rcv Err:Bad Input Arg:BufPtr 0x%x,pipe %d,t/o %d,app %s'
```

Global CFE_SB_SEND_BAD_ARG_EID

```
'Send Err:Bad input argument,Arg 0x%x,App %s'
```

Global CFE_SB_SEND_INV_MSGID_EID

```
'Send Err:Invalid msgid in msg,MsgId 0x%x,App %s'
```

Global CFE_SB_SEND_NO_SUBS_EID

```
'No subscribers for MsgId 0x%x,sender %s'
```

Global CFE_SB_SETPIPEOPTS_EID

```
'SetPipeOpts:  Options set (%d).  app %s'
```

Global CFE_SB_SETPIPEOPTS_ID_ERR_EID

```
'SetPipeOptsErr:Invalid pipe id (%d).app %s'
```

Global CFE_SB_SETPIPEOPTS_OWNER_ERR_EID

```
'SetPipeOptsErr:Caller not owner (%d).app %s'
```

Global CFE_SB_SND_RTG_EID

```
'%s written:Size=%d,Entries=%d'
```

Global CFE_SB_SND_RTG_ERR1_EID

```
'Error creating file %s, stat=0x%x'
```

Global CFE_SB_SND_STATS_EID

```
'Software Bus Statistics packet sent'
```

Global CFE_SB_SUB_ARG_ERR_EID

```
'Subscribe Err:Bad Arg,MsgId 0x%x,PipeId %d,app %s,scope %d'
```

Global CFE_SB_SUB_INV_CALLER_EID

```
'Subscribe Err:Caller(%s) is not the owner of pipe %d, Msg=0x%x'
```

Global CFE_SB_SUB_INV_PIPE_EID

```
'Subscribe Err:Invalid Pipe Id,Msg=0x%x,PipeId=%d,App %s'
```

Global CFE_SB_SUBSCRIPTION_RCVD_EID

```
'Subscription Rcvd:MsgId 0x%x on %s(%d),app %s'
```

Global CFE_SB_SUBSCRIPTION_REMOVED_EID

'Subscription Removed:Msg 0x%x on pipe %d,app %s'

Global CFE_SB_SUBSCRIPTION_RPT_EID

'Sending Subscription Report Msg=0x%x,Pipe=%d,Stat=0x%x'

Global CFE_SB_UNSUB_ARG_ERR_EID

'UnSubscribe Err:Bad Arg,MsgId 0x%x,PipeId %d,app %s,scope %d'

Global CFE_SB_UNSUB_INV_CALLER_EID

'Unsubscribe Err:Caller(%s) is not the owner of pipe %d,Msg=0x%x'

Global CFE_SB_UNSUB_INV_PIPE_EID

'Unsubscribe Err:Invalid Pipe Id Msg=0x%x,Pipe=%d,app=%s'

Global CFE_SB_UNSUB_NO_SUBS_EID

'Unsubscribe Err:No subs for Msg 0x%x on %s,app %s'

Global CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID

'Illegal attempt to activate dump-only table '%s''

Global CFE_TBL_ACTIVATE_ERR_EID

'Cannot activate table '%s'. No Inactive image available'

Global CFE_TBL_ASSUMED_VALID_INF_EID

'Tbl Services assumes '%s' is valid. No Validation Function has been registered'

Global CFE_TBL_CC1_ERR_EID

'Invalid command code - ID = 0x%X, CC = %d'

Global CFE_TBL_CDS_DELETE_ERR_EID

'Error while deleting '%s' from CDS, See SysLog.(Err=0x%08X)'

Global CFE_TBL_CDS_DELETED_INFO_EID

'Successfully removed '%s' from CDS'

Global CFE_TBL_CDS_NOT_FOUND_ERR_EID

'Unable to locate '%s' in CDS Registry'

Global CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID

'CDS '%s' owning app is still active'

Global CFE_TBL_CREATING_DUMP_FILE_ERR_EID

'Error creating dump file '%s', Status=0x%08X'

Global CFE_TBL_DUMP_PENDING_ERR_EID

'A dump for '%s' is already pending'

Global CFE_TBL_FAIL_HK_SEND_ERR_EID

'Unable to send Hk Packet (Status=0x%08X)'

Global CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID

'Manage Notification Pkt Error(MsgId=0x%08X, CC=0x%04X, Param=0x%08X, Status=0x%08X)'

Global CFE_TBL_FILE_ACCESS_ERR_EID

'Unable to open file '%s' for table load, Status = 0x%08X'

Global CFE_TBL_FILE_INCOMPLETE_ERR_EID

'Incomplete load of '%s' into '%s' working buffer'

Global CFE_TBL_FILE_LOADED_INF_EID

'Successful load of '%s' into '%s' working buffer'

Global CFE_TBL_FILE_STD_HDR_ERR_EID

'Unable to read std header for '%s', Status = 0x%08X'

Global CFE_TBL_FILE_SUBTYPE_ERR_EID

'File subtype for '%s' is wrong. Subtype = 0x%08X'

Global CFE_TBL_FILE_TBL_HDR_ERR_EID

'Unable to read tbl header for '%s', Status = 0x%08X'

Global CFE_TBL_FILE_TOO_BIG_ERR_EID

'File '%s' has more data than Tbl Hdr indicates (%d)'

Global CFE_TBL_FILE_TYPE_ERR_EID

'File '%s' is not a cFE file type, ContentType = 0x%08X'

Global CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID

'Cmd for Table '%s' had illegal buffer parameter (0x%08X)'

Global CFE_TBL_IN_REGISTRY_ERR_EID

''%s' found in Table Registry. CDS cannot be deleted until table is unregistered'

Global CFE_TBL_INIT_INF_EID

'Task Initialized'

Global CFE_TBL_INTERNAL_ERROR_ERR_EID

'Internal Error (Status=0x%08X)'

Global CFE_TBL_LEN_ERR_EID

'Invalid cmd pkt - ID = 0x%X, CC = %d, Len = %d'

Global CFE_TBL_LOAD_ABORT_ERR_EID

'Cannot abort load of '%s'. No load started.'

Global CFE_TBL_LOAD_ABORT_INF_EID

'Table Load Aborted for '%s''

Global CFE_TBL_LOAD_ERR_EID

'%s Failed to Load '%s' from %s, Status=0x%08X" </tt></dd> <dt>\anchor _↩
cfeevents000235 Global _internalref cfe__tbl__events_8h#ad2081d33addd3a6296e76dbc2b049b
"CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID" </dt><dd> <tt> 'Cannot load '\%s' (\%d)
at offset \%d in '\%s' (\%d)' </tt></dd> <dt>\anchor _cfeevents000208 ↩
Global _internalref cfe__tbl__events_8h#aae47be6124d1c76374510ddb181ce2da
"CFE_TBL_LOAD_PEND_REQ_INF_EID" </dt><dd> <tt> 'Tbl Services notifying App
that '\%s' has a load pending' </tt></dd> <dt>\anchor _cfeevents000253 ↩
Global _internalref cfe__tbl__events_8h#a5a321b08d40bf14dd5e772058d617609
"CFE_TBL_LOAD_SUCCESS_INF_EID" </dt><dd> <tt> 'Successfully loaded '\%s'
from '\%s'' </tt></dd> <dt>\anchor _cfeevents000261 Global _internalref
cfe__tbl__events_8h#a654ba428e965a9cf401edf6697a2075c "CFE_TBL_LOAD_TYPE_E↩
RR_EID" </dt><dd> <tt> '\%s Failed to Load '\%s' (Invalid Source Type)"

Global CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID

'Attempted to load DUMP-ONLY table '%s' from '%s''

Global CFE_TBL_LOADING_PENDING_ERR_EID

'Attempted to load table '%s' while previous load is still pending'

Global CFE_TBL_MID_ERR_EID

'Invalid message ID - ID = 0x%X'

Global CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID

'No Inactive Buffer for Table '%s' present'

Global CFE_TBL_NO_SUCH_TABLE_ERR_EID

'Unable to locate '%s' in Table Registry'

Global CFE_TBL_NO_WORK_BUFFERS_ERR_EID

'No working buffers available for table '%s''

Global CFE_TBL_NOOP_INF_EID

'No-op command'

Global CFE_TBL_NOT_CRITICAL_TBL_ERR_EID

'Table '%s' is in Critical Table Registry but CDS is not tagged as a table'

Global CFE_TBL_NOT_IN_CRIT_REG_ERR_EID

'Table '%s' is not found in Critical Table Registry'

Global CFE_TBL_OVERWRITE_DUMP_INF_EID

'Successfully overwrote '%s' with Table '%s''

Global CFE_TBL_OVERWRITE_REG_DUMP_INF_EID

'Successfully overwrote '%s' with Table Registry'

Global CFE_TBL_PARTIAL_LOAD_ERR_EID

''%s' has partial load for uninitialized table '%s''

Global CFE_TBL_PROCESSOR_ID_ERR_EID

'Unable to verify Processor ID for '%s', ID = 0x%08X'

Global CFE_TBL_REGISTER_ERR_EID

'%s Failed to Register '%s', Status=0x%08X'

Global CFE_TBL_RESET_INF_EID

'Reset Counters command'

Global CFE_TBL_SHARE_ERR_EID

'%s Failed to Share '%s', Status=0x%08X'

Global CFE_TBL_SPACECRAFT_ID_ERR_EID

'Unable to verify Spacecraft ID for '%s', ID = 0x%08X'

Global CFE_TBL_TLM_REG_CMD_INF_EID

'Table Registry entry for '%s' will be telemetered'

Global CFE_TBL_TOO_MANY_DUMPS_ERR_EID

'Too many Dump Only Table Dumps have been requested'

Global CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID

'Too many Table Validations have been requested'

Global CFE_TBL_UNREGISTER_ERR_EID

'%s Failed to Unregister '%s', Status=0x%08X'

Global CFE_TBL_UNVALIDATED_ERR_EID

'Cannot activate table '%s'. Inactive image not Validated'

Global CFE_TBL_UPDATE_ERR_EID

'%s Failed to Update '%s', Status=0x%08X" </tt></dd> <dt>\anchor _cfeevents000255 Global _internalref cfe__tbl__events_8h#ae29dd1189f2b5cd66597707155b66463 "CFE_TBL_UPDATE_SUCCESS_INF_EID" </dt><dd> <tt> '\%s Successfully Updated '\%s'' </tt></dd> <dt>\anchor _cfeevents000207 Global _internalref cfe_↩ _tbl__events_8h#aee127ace865d00d583dfdf151ed12568 "CFE_TBL_VAL_REQ_MADE_I↩ NF_EID" </dt><dd> <tt> 'Tbl Services issued validation request for '\%s'' </tt></dd> <dt>\anchor _cfeevents000263 Global _internalref cfe__tbl_↩ _events_8h#aee9c1716b5b0b451f2d0bfc0e247a262 "CFE_TBL_VALIDATION_ERR_EID" </dt><dd> <tt> '\%s validation failed for Inactive '\%s', Status=0x\%08X"

Global CFE_TBL_VALIDATION_INF_EID

'%s validation successful for Inactive '%s''

Global CFE_TBL_WRITE_CFE_HDR_ERR_EID

'Error writing cFE File Header to '%s', Status=0x%08X'

Global CFE_TBL_WRITE_DUMP_INF_EID

'Successfully dumped Table '%s' to '%s''

Global CFE_TBL_WRITE_REG_DUMP_INF_EID

'Successfully dumped Table Registry to '%s':Size=%d,Entries=%d'

Global CFE_TBL_WRITE_TBL_HDR_ERR_EID

'Error writing Tbl image File Header to '%s', Status=0x%08X'

Global CFE_TBL_WRITE_TBL_IMG_ERR_EID

'Error writing Tbl image to '%s', Status=0x%08X'

Global CFE_TBL_WRITE_TBL_REG_ERR_EID

'Error writing Registry to '%s', Status=0x%08X'

Global CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID

'Table Hdr in '%s' indicates no data in file'

Global CFE_TIME_1HZ_CFG_EID

'1Hz Adjust commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'

Global CFE_TIME_1HZ_EID

'STCF 1Hz Adjust - secs = %d, ssecs = 0x%X, dir = %d'

Global CFE_TIME_CC_ERR_EID

'Invalid command code - ID = 0x%X, CC = %d'

Global CFE_TIME_DELAY_CFG_EID

'Set Delay commands invalid without CFE_PLATFORM_TIME_CFG_CLIENT set to true'

Global CFE_TIME_DELAY_EID

'Set Tone Delay - secs = %d, usecs = %d, ssecs = 0x%X, dir = %d'

Global CFE_TIME_DELAY_ERR_EID

'Invalid Tone Delay - secs = %d, usecs = %d'

Global CFE_TIME_DELTA_CFG_EID

'STCF Adjust commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'

Global CFE_TIME_DELTA_EID

'STCF Adjust - secs = %d, usecs = %d, ssecs = 0x%X, dir[1=Positive, 2=Negative] = %d'

Global CFE_TIME_DELTA_ERR_EID

'Invalid STCF Adjust - secs = %d, usecs = %d, dir[1=Positive, 2=Negative] = %d'

Global CFE_TIME_DIAG_EID

'Request diagnostics command'

Global CFE_TIME_FLY_OFF_EID

'Stop FLYWHEEL'

Global CFE_TIME_FLY_ON_EID

'Start FLYWHEEL'

Global CFE_TIME_ID_ERR_EID

'Invalid message ID - ID = 0x%X'

Global CFE_TIME_INIT_EID

'cFE TIME Initialized'

Global CFE_TIME_LEAPS_CFG_EID

'Set Leaps commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'

Global CFE_TIME_LEAPS_EID

'Set Leap Seconds = %d'

Global CFE_TIME_LEN_ERR_EID

'Invalid cmd length: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'

Global CFE_TIME_MET_CFG_EID

'Set MET commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'

Global CFE_TIME_MET_EID

'Set MET - secs = %d, usecs = %d, ssecs = 0x%X'

Global CFE_TIME_MET_ERR_EID

'Invalid MET - secs = %d, usecs = %d'

Global CFE_TIME_NOOP_EID

'No-op command'

Global CFE_TIME_RESET_EID

'Reset Counters command'

Global CFE_TIME_SIGNAL_CFG_EID

'Set Signal commands invalid without CFE_PLATFORM_TIME_CFG_SIGNAL set to true'

Global CFE_TIME_SIGNAL_EID

'Set Tone Source = %s'

Global CFE_TIME_SIGNAL_ERR_EID

'Invalid Tone Source = 0x%X'

Global CFE_TIME_SOURCE_CFG_EID

'Set Source commands invalid without CFE_PLATFORM_TIME_CFG_SOURCE set to true'

Global CFE_TIME_SOURCE_EID

'Set Time Source = %s'

Global CFE_TIME_SOURCE_ERR_EID

```
'Invalid Time Source = 0x%X'
```

Global CFE_TIME_STATE_EID

```
'Set Clock State = %s'
```

Global CFE_TIME_STATE_ERR_EID

```
'Invalid Clock State = 0x%X'
```

Global CFE_TIME_STCF_CFG_EID

```
'Set STCF commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
```

Global CFE_TIME_STCF_EID

```
'Set STCF - secs = %d, usecs = %d, ssecs = 0x%X'
```

Global CFE_TIME_STCF_ERR_EID

```
'Invalid STCF - secs = %d, usecs = %d'
```

Global CFE_TIME_TIME_CFG_EID

```
'Set Time commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
```

Global CFE_TIME_TIME_EID

```
'Set Time - secs = %d, usecs = %d, ssecs = 0x%X'
```

Global CFE_TIME_TIME_ERR_EID

```
'Invalid Time - secs = %d, usecs = %d'
```

8.24 cFE Command Mnemonic Cross Reference

The following cross reference maps the cFE command codes to Command Mnemonics. To learn about the details of a particular command, click on its associated command code.

Global CFE_ES_CLEAR_ER_LOG_CC

$sc_$cpu_ES_ClearERLog

Global CFE_ES_CLEAR_SYSLOG_CC

$sc_$cpu_ES_ClearSysLog

Global CFE_ES_DELETE_CDS_CC

$sc_$cpu_ES_DeleteCDS

Global CFE_ES_DUMP_CDS_REGISTRY_CC

$sc_$cpu_ES_WriteCDS2File

Global CFE_ES_NOOP_CC

$sc_$cpu_ES_NOOP

Global CFE_ES_OVER_WRITE_SYSLOG_CC

$sc_$cpu_ES_OverwriteSysLogMode

Global CFE_ES_QUERY_ALL_CC

$sc_$cpu_ES_WriteAppInfo2File

Global CFE_ES_QUERY_ALL_TASKS_CC

$sc_$cpu_ES_WriteTaskInfo2File

Global CFE_ES_QUERY_ONE_CC

$sc_$cpu_ES_QueryApp

Global CFE_ES_RELOAD_APP_CC
$sc_$cpu_ES_ReloadApp

Global CFE_ES_RESET_COUNTERS_CC
$sc_$cpu_ES_ResetCtrs

Global CFE_ES_RESET_PR_COUNT_CC
$sc_$cpu_ES_ResetPRCnt

Global CFE_ES_RESTART_APP_CC
$sc_$cpu_ES_ResetApp

Global CFE_ES_RESTART_CC
$sc_$cpu_ES_ProcessorReset, $sc_$cpu_ES_PowerOnReset

Global CFE_ES_SEND_MEM_POOL_STATS_CC
$sc_$cpu_ES_PoolStats

Global CFE_ES_SET_MAX_PR_COUNT_CC
$sc_$cpu_ES_SetMaxPRCnt

Global CFE_ES_SET_PERF_FILTER_MASK_CC
$sc_$cpu_ES_LAFilterMask

Global CFE_ES_SET_PERF_TRIGGER_MASK_CC
$sc_$cpu_ES_LATriggerMask

Global CFE_ES_SHELL_CC
$sc_$cpu$_ES_Shell

Global CFE_ES_START_APP_CC
$sc_$cpu_ES_StartApp

Global CFE_ES_START_PERF_DATA_CC
$sc_$cpu_ES_StartLAData

Global CFE_ES_STOP_APP_CC
$sc_$cpu_ES_StopApp

Global CFE_ES_STOP_PERF_DATA_CC
$sc_$cpu_ES_StopLAData

Global CFE_ES_WRITE_ER_LOG_CC
$sc_$cpu_ES_WriteERLog2File

Global CFE_ES_WRITE_SYSLOG_CC
$sc_$cpu_ES_WriteSysLog2File

Global CFE_EVS_ADD_EVENT_FILTER_CC
$sc_$cpu_EVS_AddEvtFltr

Global CFE_EVS_CLEAR_LOG_CC
$sc_$cpu_EVS_ClrLog

Global CFE_EVS_DELETE_EVENT_FILTER_CC
$sc_$cpu_EVS_DelEvtFltr

Global CFE_EVS_DISABLE_APP_EVENT_TYPE_CC
$sc_$cpu_EVS_DisAppEvtType, $sc_$cpu_EVS_DisAppEvtTypeMask

Global CFE_EVS_DISABLE_APP_EVENTS_CC
$sc_$cpu_EVS_DisAppEvGen

Global CFE_SB_RESET_COUNTERS_CC
$sc_$cpu_SB_ResetCtrs

Global CFE_SB_SEND_MAP_INFO_CC
$sc_$cpu_SB_WriteMap2File

Global CFE_SB_SEND_PIPE_INFO_CC
$sc_$cpu_SB_WritePipe2File

Global CFE_SB_SEND_PREV_SUBS_CC
$sc_$cpu_SB_SendPrevSubs

Global CFE_SB_SEND_ROUTING_INFO_CC
$sc_$cpu_SB_WriteRouting2File

Global CFE_SB_SEND_SB_STATS_CC
$sc_$cpu_SB_DumpStats

Global CFE_TBL_ABORT_LOAD_CC
$sc_$cpu_TBL_LOADABORT

Global CFE_TBL_ACTIVATE_CC
$sc_$cpu_TBL_ACTIVATE

Global CFE_TBL_DELETE_CDS_CC
$sc_$cpu_TBL_DeleteCDS

Global CFE_TBL_DUMP_CC
$sc_$cpu_TBL_DUMP

Global CFE_TBL_DUMP_REGISTRY_CC
$sc_$cpu_TBL_WriteReg2File

Global CFE_TBL_LOAD_CC
$sc_$cpu_TBL_Load

Global CFE_TBL_NOOP_CC
$sc_$cpu_TBL_NOOP

Global CFE_TBL_RESET_COUNTERS_CC
$sc_$cpu_TBL_ResetCtrs

Global CFE_TBL_SEND_REGISTRY_CC
$sc_$cpu_TBL_TLMReg

Global CFE_TBL_VALIDATE_CC
$sc_$cpu_TBL_VALIDATE

Global CFE_TIME_ADD_1HZ_ADJUSTMENT_CC
$sc_$cpu_TIME_Add1HzSTCF

Global CFE_TIME_ADD_ADJUST_CC
$sc_$cpu_TIME_AddSTCFAdj

Global CFE_TIME_ADD_DELAY_CC
$sc_$cpu_TIME_AddClockLat

Global CFE_TIME_NOOP_CC
$sc_$cpu_TIME_NOOP

Global CFE_TIME_RESET_COUNTERS_CC
$sc_$cpu_TIME_ResetCtrs

Global CFE_TIME_SEND_DIAGNOSTIC_TLM_CC
`$sc_$cpu_TIME_RequestDiag`

Global CFE_TIME_SET_LEAP_SECONDS_CC
`$sc_$cpu_TIME_SetClockLeap`

Global CFE_TIME_SET_MET_CC
`$sc_$cpu_TIME_SetClockMET`

Global CFE_TIME_SET_SIGNAL_CC
`$sc_$cpu_TIME_SetSignal`

Global CFE_TIME_SET_SOURCE_CC
`$sc_$cpu_TIME_SetSource`

Global CFE_TIME_SET_STATE_CC
`$sc_$cpu_TIME_SetState`

Global CFE_TIME_SET_STCF_CC
`$sc_$cpu_TIME_SetClockSTCF`

Global CFE_TIME_SET_TIME_CC
`$sc_$cpu_TIME_SetClock`

Global CFE_TIME_SUB_1HZ_ADJUSTMENT_CC
`$sc_$cpu_TIME_Sub1HzSTCF`

Global CFE_TIME_SUB_ADJUST_CC
`$sc_$cpu_TIME_SubSTCFAdj`

Global CFE_TIME_SUB_DELAY_CC
`$sc_$cpu_TIME_SubClockLat`

8.25 cFE Telemetry Mnemonic Cross Reference

The following cross reference maps the cFE telemetry packet members to their associated ground system telemetry mnemonics.

Global CFE_ES_AppInfo_t::AddressesAreValid
`$sc_$cpu_ES_AddrsValid`

Global CFE_ES_AppInfo_t::AppId
`$sc_$cpu_ES_AppID`

Global CFE_ES_AppInfo_t::BSSAddress
`$sc_$cpu_ES_BSSAddress`

Global CFE_ES_AppInfo_t::BSSSize
`$sc_$cpu_ES_BSSSize`

Global CFE_ES_AppInfo_t::CodeAddress
`$sc_$cpu_ES_CodeAddress`

Global CFE_ES_AppInfo_t::CodeSize
`$sc_$cpu_ES_CodeSize`

Global CFE_ES_AppInfo_t::DataAddress
`$sc_$cpu_ES_DataAddress`

Global CFE_ES_AppInfo_t::DataSize

`$sc_$cpu_ES_DataSize`

Global CFE_ES_AppInfo_t::EntryPoint [OS_MAX_API_NAME]

`$sc_$cpu_ES_AppEntryPt`[OS_MAX_API_NAME]

Global CFE_ES_AppInfo_t::ExceptionAction

`$sc_$cpu_ES_ExceptnActn`

Global CFE_ES_AppInfo_t::ExecutionCounter

`$sc_$cpu_ES_ExecutionCtr`

Global CFE_ES_AppInfo_t::FileName [OS_MAX_PATH_LEN]

`$sc_$cpu_ES_AppFilename`[OS_MAX_PATH_LEN]

Global CFE_ES_AppInfo_t::MainTaskId

`$sc_$cpu_ES_MainTaskId`

Global CFE_ES_AppInfo_t::MainTaskName [OS_MAX_API_NAME]

`$sc_$cpu_ES_MainTaskName`[OS_MAX_API_NAME]

Global CFE_ES_AppInfo_t::ModuleId

`$sc_$cpu_ES_ModuleID`

Global CFE_ES_AppInfo_t::Name [OS_MAX_API_NAME]

`$sc_$cpu_ES_AppName`[OS_MAX_API_NAME]

Global CFE_ES_AppInfo_t::NumOfChildTasks

`$sc_$cpu_ES_ChildTasks`

Global CFE_ES_AppInfo_t::Priority

`$sc_$cpu_ES_Priority`

Global CFE_ES_AppInfo_t::StackSize

`$sc_$cpu_ES_StackSize`

Global CFE_ES_AppInfo_t::StartAddress

`$sc_$cpu_ES_StartAddr`

Global CFE_ES_AppInfo_t::Type

`$sc_$cpu_ES_AppType`

Global CFE_ES_HousekeepingTlm_Payload_t::BootSource

`$sc_$cpu_ES_BootSource`

Global CFE_ES_HousekeepingTlm_Payload_t::CFECoreChecksum

`$sc_$cpu_ES_CKSUM`

Global CFE_ES_HousekeepingTlm_Payload_t::CFEMajorVersion

`$sc_$cpu_ES_CFEMAJORVER`

Global CFE_ES_HousekeepingTlm_Payload_t::CFEMinorVersion

`$sc_$cpu_ES_CFEMINORVER`

Global CFE_ES_HousekeepingTlm_Payload_t::CFEMissionRevision

`$sc_$cpu_ES_CFEMISSIONREV`

Global CFE_ES_HousekeepingTlm_Payload_t::CFERevision

`$sc_$cpu_ES_CFEREVISION`

Global CFE_ES_HousekeepingTlm_Payload_t::CommandCounter

`$sc_$cpu_ES_CMDPC`

Global CFE_ES_HousekeepingTlm_Payload_t::RegisteredCoreApps
`$sc_$cpu_ES_RegCoreApps`

Global CFE_ES_HousekeepingTlm_Payload_t::RegisteredExternalApps
`$sc_$cpu_ES_RegExtApps`

Global CFE_ES_HousekeepingTlm_Payload_t::RegisteredLibs
`$sc_$cpu_ES_RegLibs`

Global CFE_ES_HousekeepingTlm_Payload_t::RegisteredTasks
`$sc_$cpu_ES_RegTasks`

Global CFE_ES_HousekeepingTlm_Payload_t::ResetSubtype
`$sc_$cpu_ES_ResetSubtype`

Global CFE_ES_HousekeepingTlm_Payload_t::ResetType
`$sc_$cpu_ES_ResetType`

Global CFE_ES_HousekeepingTlm_Payload_t::SysLogBytesUsed
`$sc_$cpu_ES_SYSLOGBYTEUSED`

Global CFE_ES_HousekeepingTlm_Payload_t::SysLogEntries
`$sc_$cpu_ES_SYSLOGENTRIES`

Global CFE_ES_HousekeepingTlm_Payload_t::SysLogMode
`$sc_$cpu_ES_SYSLOGMODE`

Global CFE_ES_HousekeepingTlm_Payload_t::SysLogSize
`$sc_$cpu_ES_SYSLOGSIZE`

Global CFE_ES_MemPoolStats_t::BlockStats [CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]
`$sc_$cpu_ES_BlkStats`[BLK_SIZES]

Global CFE_ES_MemPoolStats_t::CheckErrCtr
`$sc_$cpu_ES_BlkErrCTR`

Global CFE_ES_MemPoolStats_t::NumBlocksRequested
`$sc_$cpu_ES_BlksREQ`

Global CFE_ES_MemPoolStats_t::NumFreeBytes
`$sc_$cpu_ES_FreeBytes`

Global CFE_ES_MemPoolStats_t::PoolSize
`$sc_$cpu_ES_PoolSize`

Global CFE_ES_PoolStatsTlm_Payload_t::PoolHandle
`$sc_$cpu_ES_PoolHandle`

Global CFE_EVS_AppTlmData_t::AppEnableStatus
`$sc_$cpu_EVS_APP`[CFE_ES_MAX_APPLICATIONS].APPENASTAT

Global CFE_EVS_AppTlmData_t::AppID
`$sc_$cpu_EVS_APP`[CFE_ES_MAX_APPLICATIONS].APPID

Global CFE_EVS_AppTlmData_t::AppMessageSentCounter
`$sc_$cpu_EVS_APP`[CFE_ES_MAX_APPLICATIONS].APPMSGSENTC

Global CFE_EVS_AppTlmData_t::Padding
`$sc_$cpu_EVS_APP`[CFE_ES_MAX_APPLICATIONS].SPARE2ALIGN3

Global CFE_EVS_HousekeepingTlm_Payload_t::AppData [CFE_MISSION_ES_MAX_APPLICATIONS]
`$sc_$cpu_EVS_APP`[CFE_ES_MAX_APPLICATIONS]

Global CFE_EVS_HousekeepingTlm_Payload_t::CommandCounter
$sc_$cpu_EVS_CMDPC

Global CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter
$sc_$cpu_EVS_CMDEC

Global CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled
$sc_$cpu_EVS_LOGENABLED

Global CFE_EVS_HousekeepingTlm_Payload_t::LogFullFlag
$sc_$cpu_EVS_LOGFULL

Global CFE_EVS_HousekeepingTlm_Payload_t::LogMode
$sc_$cpu_EVS_LOGMODE

Global CFE_EVS_HousekeepingTlm_Payload_t::LogOverflowCounter
$sc_$cpu_EVS_LOGOVERFLOWC

Global CFE_EVS_HousekeepingTlm_Payload_t::MessageFormatMode
$sc_$cpu_EVS_MSGFMTMODE

Global CFE_EVS_HousekeepingTlm_Payload_t::MessageSendCounter
$sc_$cpu_EVS_MSGSENTC

Global CFE_EVS_HousekeepingTlm_Payload_t::MessageTruncCounter
$sc_$cpu_EVS_MSGTRUNC

Global CFE_EVS_HousekeepingTlm_Payload_t::OutputPort
$sc_$cpu_EVS_OUTPUTPORT

Global CFE_EVS_HousekeepingTlm_Payload_t::Spare1
$sc_$cpu_EVS_HK_SPARE1

Global CFE_EVS_HousekeepingTlm_Payload_t::Spare2
$sc_$cpu_EVS_HK_SPARE2

Global CFE_EVS_HousekeepingTlm_Payload_t::Spare3
$sc_$cpu_EVS_HK_SPARE3

Global CFE_EVS_HousekeepingTlm_Payload_t::UnregisteredAppCounter
$sc_$cpu_EVS_UNREGAPPC

Global CFE_EVS_LongEventTlm_Payload_t::Message [CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]
$sc_$cpu_EVS_EVENT[CFE_EVS_MAX_MESSAGE_LENGTH]

Global CFE_EVS_LongEventTlm_Payload_t::Spare1
$sc_$cpu_EVS_SPARE1

Global CFE_EVS_LongEventTlm_Payload_t::Spare2
$sc_$cpu_EVS_SPARE2

Global CFE_EVS_PacketID_t::AppName [CFE_MISSION_MAX_API_LEN]
$sc_$cpu_EVS_APPNAME[OS_MAX_API_NAME]

Global CFE_EVS_PacketID_t::EventID
$sc_$cpu_EVS_EVENTID

Global CFE_EVS_PacketID_t::EventType
$sc_$cpu_EVS_EVENTTYPE

Global CFE_EVS_PacketID_t::ProcessorID
$sc_$cpu_EVS_PROCESSORID

Global CFE_EVS_PacketID_t::SpacecraftID
`$sc_$cpu_EVS_SCID`

Global CFE_SB_HousekeepingTlm_Payload_t::CommandCounter
`$sc_$cpu_SB_CMDPC`

Global CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter
`$sc_$cpu_SB_CMDEC`

Global CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter
`$sc_$cpu_SB_NewPipeEC`

Global CFE_SB_HousekeepingTlm_Payload_t::DuplicateSubscriptionsCounter
`$sc_$cpu_SB_DupSubCnt`

Global CFE_SB_HousekeepingTlm_Payload_t::GetPipeIdByNameErrorCounter

Global CFE_SB_HousekeepingTlm_Payload_t::InternalErrorCounter
`$sc_$cpu_SB_InternalEC`

Global CFE_SB_HousekeepingTlm_Payload_t::MemInUse
`$sc_$cpu_SB_MemInUse`

Global CFE_SB_HousekeepingTlm_Payload_t::MemPoolHandle
`$sc_$cpu_SB_MemPoolHdl`

Global CFE_SB_HousekeepingTlm_Payload_t::MsgLimitErrorCounter
`$sc_$cpu_SB_MsgLimEC`

Global CFE_SB_HousekeepingTlm_Payload_t::MsgReceiveErrorCounter
`$sc_$cpu_SB_MsgRecEC`

Global CFE_SB_HousekeepingTlm_Payload_t::MsgSendErrorCounter
`$sc_$cpu_SB_MsgSndEC`

Global CFE_SB_HousekeepingTlm_Payload_t::NoSubscribersCounter
`$sc_$cpu_SB_NoSubEC`

Global CFE_SB_HousekeepingTlm_Payload_t::PipeOptsErrorCounter

Global CFE_SB_HousekeepingTlm_Payload_t::PipeOverflowErrorCounter
`$sc_$cpu_SB_PipeOvrEC`

Global CFE_SB_HousekeepingTlm_Payload_t::Spare2Align [1]
`$sc_$cpu_SB_Spare2Align[2]`

Global CFE_SB_HousekeepingTlm_Payload_t::SubscribeErrorCounter
`$sc_$cpu_SB_SubscrEC`

Global CFE_SB_HousekeepingTlm_Payload_t::UnmarkedMem
`$sc_$cpu_SB_UnMarkedMem`

Global CFE_SB_PipeDepthStats_t::Depth
`$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES].SB_PDDEPTH`

Global CFE_SB_PipeDepthStats_t::InUse
`$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES].SB_PDINUSE`

Global CFE_SB_PipeDepthStats_t::PeakInUse
`$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES].SB_PDPKINUSE`

Global CFE_SB_PipeDepthStats_t::PipeId

`$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES].SB_PDPIPEID`

Global CFE_SB_PipeDepthStats_t::Spare

`$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES].SB_PDSPARE`

Global CFE_SB_StatsTlm_Payload_t::MaxMemAllowed

`$sc_$cpu_SB_Stat.SB_SMMBMALW`

Global CFE_SB_StatsTlm_Payload_t::MaxMsgIdsAllowed

`$sc_$cpu_SB_Stat.SB_SMMMIDALW`

Global CFE_SB_StatsTlm_Payload_t::MaxPipeDepthAllowed

`$sc_$cpu_SB_Stat.SB_SMMPDALW`

Global CFE_SB_StatsTlm_Payload_t::MaxPipesAllowed

`$sc_$cpu_SB_Stat.SB_SMMPALW`

Global CFE_SB_StatsTlm_Payload_t::MaxSubscriptionsAllowed

`$sc_$cpu_SB_Stat.SB_SMMSALW`

Global CFE_SB_StatsTlm_Payload_t::MemInUse

`$sc_$cpu_SB_Stat.SB_SMBMIU`

Global CFE_SB_StatsTlm_Payload_t::MsgIdsInUse

`$sc_$cpu_SB_Stat.SB_SMMIDIU`

Global CFE_SB_StatsTlm_Payload_t::PeakMemInUse

`$sc_$cpu_SB_Stat.SB_SMPBMIU`

Global CFE_SB_StatsTlm_Payload_t::PeakMsgIdsInUse

`$sc_$cpu_SB_Stat.SB_SMPMIDIU`

Global CFE_SB_StatsTlm_Payload_t::PeakPipesInUse

`$sc_$cpu_SB_Stat.SB_SMPPIU`

Global CFE_SB_StatsTlm_Payload_t::PeakSBBuffersInUse

`$sc_$cpu_SB_Stat.SB_SMPSBBIU`

Global CFE_SB_StatsTlm_Payload_t::PeakSubscriptionsInUse

`$sc_$cpu_SB_Stat.SB_SMPSIU`

Global CFE_SB_StatsTlm_Payload_t::PipeDepthStats [CFE_MISSION_SB_MAX_PIPES]

`$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]`

Global CFE_SB_StatsTlm_Payload_t::PipesInUse

`$sc_$cpu_SB_Stat.SB_SMPIU`

Global CFE_SB_StatsTlm_Payload_t::SBBuffersInUse

`$sc_$cpu_SB_Stat.SB_SMSBBIU`

Global CFE_SB_StatsTlm_Payload_t::SubscriptionsInUse

`$sc_$cpu_SB_Stat.SB_SMSIU`

Global CFE_TBL_HousekeepingTlm_Payload_t::ActiveBuffer

`$sc_$cpu_TBL_LastValBuf`

Global CFE_TBL_HousekeepingTlm_Payload_t::ByteAlignPad1

`$sc_$cpu_TBL_ByteAlignPad1`

Global CFE_TBL_HousekeepingTlm_Payload_t::CommandCounter

`$sc_$cpu_TBL_CMDPC`

Global CFE_TBL_HousekeepingTlm_Payload_t::CommandErrorCounter

$sc_$cpu_TBL_CMDEC

Global CFE_TBL_HousekeepingTlm_Payload_t::FailedValCounter

$sc_$cpu_TBL_ValFailedCtr

Global CFE_TBL_HousekeepingTlm_Payload_t::LastFileDumped [CFE_MISSION_MAX_PATH_LEN]

$sc_$cpu_TBL_LastFileDumped[OS_MAX_PATH_LEN]

Global CFE_TBL_HousekeepingTlm_Payload_t::LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]

$sc_$cpu_TBL_LastFileLoaded[OS_MAX_PATH_LEN]

Global CFE_TBL_HousekeepingTlm_Payload_t::LastTableLoaded [CFE_MISSION_TBL_MAX_FULL_NAME_L↩EN]

$sc_$cpu_TBL_LastTableLoaded[CFE_TBL_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload_t::LastUpdatedTable [CFE_MISSION_TBL_MAX_FULL_NAME_↩LEN]

$sc_$cpu_TBL_LastUpdTblName[CFE_TB_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload_t::LastUpdateTime

$sc_$cpu_TBL_LastUpdTime, $sc_$cpu_TBL_SECONDS, $sc_$cpu_TBL_SUBSECONDS

Global CFE_TBL_HousekeepingTlm_Payload_t::LastValCrc

$sc_$cpu_TBL_LastValCRC

Global CFE_TBL_HousekeepingTlm_Payload_t::LastValStatus

$sc_$cpu_TBl_LastValS

Global CFE_TBL_HousekeepingTlm_Payload_t::LastValTableName [CFE_MISSION_TBL_MAX_FULL_NAME_↩LEN]

$sc_$cpu_TBL_LastValTblName[CFE_TB_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload_t::MemPoolHandle

$sc_$cpu_TBL_MemPoolHandle

Global CFE_TBL_HousekeepingTlm_Payload_t::NumFreeSharedBufs

$sc_$cpu_TBL_NumFreeShrBuf

Global CFE_TBL_HousekeepingTlm_Payload_t::NumLoadPending

$sc_$cpu_TBL_NumUpdatesPend

Global CFE_TBL_HousekeepingTlm_Payload_t::NumTables

$sc_$cpu_TBL_NumTables

Global CFE_TBL_HousekeepingTlm_Payload_t::NumValRequests

$sc_$cpu_TBL_ValReqCtr

Global CFE_TBL_HousekeepingTlm_Payload_t::SuccessValCounter

$sc_$cpu_TBL_ValSuccessCtr

Global CFE_TBL_HousekeepingTlm_Payload_t::ValidationCounter

$sc_$cpu_TBL_ValCompltdCtr

Global CFE_TBL_TblRegPacket_Payload_t::ActiveBufferAddr

$sc_$cpu_TBL_ActBufAdd

Global CFE_TBL_TblRegPacket_Payload_t::ByteAlign4

$sc_$cpu_TBL_Spare4

Global CFE_TIME_DiagnosticTlm_Payload_t::ClockSetState

$sc_$cpu_TIME_DValid

Global CFE_TIME_DiagnosticTlm_Payload_t::ClockSignal

$sc_$cpu_TIME_DSignal

Global CFE_TIME_DiagnosticTlm_Payload_t::ClockSource

$sc_$cpu_TIME_DSource

Global CFE_TIME_DiagnosticTlm_Payload_t::ClockStateAPI

$sc_$cpu_TIME_DAPIState

Global CFE_TIME_DiagnosticTlm_Payload_t::ClockStateFlags

$sc_$cpu_TIME_DStateFlags, $sc_$cpu_TIME_DFlagSet, $sc_$cpu_TIME_DFlagFly, $sc_$cpu_TIME_DFlagSrc, $sc_$cpu_TIME_DFlagPri, $sc_$cpu_TIME_DFlagSfly, $sc_$cpu_TIME_DFlagCfly, $sc_$cpu_TIME_DFlagAdjd, $sc_$cpu_TIME_DFlag1Hzd, $sc_$cpu_TIME_DFlagClat, $sc_$cpu_TIME_DFlagSorC, $sc_$cpu_TIME_DFlagNIU

Global CFE_TIME_DiagnosticTlm_Payload_t::CurrentLatch

$sc_$cpu_TIME_DLocalS, $sc_$cpu_TIME_DLocalSs

Global CFE_TIME_DiagnosticTlm_Payload_t::CurrentMET

$sc_$cpu_TIME_DMETS, $sc_$cpu_TIME_DMETSs

Global CFE_TIME_DiagnosticTlm_Payload_t::CurrentTAI

$sc_$cpu_TIME_DTAIS, $sc_$cpu_TIME_DTAISS

Global CFE_TIME_DiagnosticTlm_Payload_t::CurrentUTC

$sc_$cpu_TIME_DUTCS, $sc_$cpu_TIME_DUTCSS

Global CFE_TIME_DiagnosticTlm_Payload_t::DataStoreStatus

$sc_$cpu_TIME_DataStStat

Global CFE_TIME_DiagnosticTlm_Payload_t::DelayDirection

$sc_$cpu_TIME_DLatentDir

Global CFE_TIME_DiagnosticTlm_Payload_t::Forced2Fly

$sc_$cpu_TIME_DCMD2Fly

Global CFE_TIME_DiagnosticTlm_Payload_t::LocalIntCounter

$sc_$cpu_TIME_D1HzISRCNT

Global CFE_TIME_DiagnosticTlm_Payload_t::LocalTaskCounter

$sc_$cpu_TIME_D1HzTaskCNT

Global CFE_TIME_DiagnosticTlm_Payload_t::MaxElapsed

$sc_$cpu_TIME_DMaxWindow

Global CFE_TIME_DiagnosticTlm_Payload_t::MaxLocalClock

$sc_$cpu_TIME_DWrapS, $sc_$cpu_TIME_DWrapSs

Global CFE_TIME_DiagnosticTlm_Payload_t::MinElapsed

$sc_$cpu_TIME_DMinWindow

Global CFE_TIME_DiagnosticTlm_Payload_t::OneHzAdjust

$sc_$cpu_TIME_D1HzAdjS, $sc_$cpu_TIME_D1HzAdjSs

Global CFE_TIME_DiagnosticTlm_Payload_t::OneHzDirection

$sc_$cpu_TIME_D1HzAdjDir

Global CFE_TIME_HousekeepingTlm_Payload_t::CommandCounter

`$sc_$cpu_TIME_CMDPC`

Global CFE_TIME_HousekeepingTlm_Payload_t::CommandErrorCounter

`$sc_$cpu_TIME_CMDEC`

Global CFE_TIME_HousekeepingTlm_Payload_t::LeapSeconds

`$sc_$cpu_TIME_LeapSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::Seconds1HzAdj

`$sc_$cpu_TIME_1HzAdjSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::SecondsDelay

`$sc_$cpu_TIME_1HzAdjSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::SecondsMET

`$sc_$cpu_TIME_METSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::SecondsSTCF

`$sc_$cpu_TIME_STCFSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::Subsecs1HzAdj

`$sc_$cpu_TIME_1HzAdjSSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::SubsecsDelay

`$sc_$cpu_TIME_1HzAdjSSecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::SubsecsMET

`$sc_$cpu_TIME_METSubsecs`

Global CFE_TIME_HousekeepingTlm_Payload_t::SubsecsSTCF

`$sc_$cpu_TIME_STCFSubsecs`

8.26 cFE Application Programmer's Interface (API) Reference

Executive Services API

- cFE Entry/Exit APIs
  - CFE_ES_Main - cFE Main Entry Point used by Board Support Package to start cFE
  - CFE_ES_ResetCFE - Reset the cFE Core and all cFE Applications
- Application Control APIs
  - CFE_ES_RestartApp - Restart a single cFE Application
  - CFE_ES_ReloadApp - Reload a single cFE Application
  - CFE_ES_DeleteApp - Delete a single cFE Application
- Application Behavior APIs
  - CFE_ES_RegisterApp - Registers a cFE Application with Executive Services
  - CFE_ES_RunLoop - Checks for Exit, Restart or Reload commands
  - CFE_ES_WaitForStartupSync - Waits for all Apps in Startup Script to complete initialization
  - CFE_ES_WaitForSystemState - Waits for minimum global system state
  - CFE_ES_IncrementTaskCounter - Increments telemetered task counter to indicate task activity

  - CFE_ES_ExitApp - Exit for a cFE Application

- Information APIs
  - CFE_ES_GetResetType - Returns the most recent type of reset
  - CFE_ES_GetAppID - Returns the Application ID of calling Application
  - CFE_ES_GetAppIDByName - Returns the Application ID of an Application specified by name
  - CFE_ES_GetAppName - Returns the name of an Application specified by Application ID
  - CFE_ES_GetAppInfo - Returns Application info about an Application specified by Application ID
  - CFE_ES_GetTaskInfo - Returns the task information about a task specified by task ID
- Child Task APIs
  - CFE_ES_RegisterChildTask - Registers a cFE Child Task with Executive Services
  - CFE_ES_CreateChildTask - Creates a new task under an existing Application
  - CFE_ES_DeleteChildTask - Deletes a task under an existing Application
  - CFE_ES_ExitChildTask - Exit for a cFE Child Task
- Critical Data Store APIs
  - CFE_ES_RegisterCDS - Reserves space (or obtains previously reserved space)in the Critical Data Store
  - CFE_ES_CopyToCDS - Copies a block of memory into a Critical Data Store
  - CFE_ES_RestoreFromCDS - Recovers the contents of memory from a Critical Data Store
- Memory Manager APIs
  - CFE_ES_PoolCreate - Initializes a Memory Pool created by an Application
  - CFE_ES_PoolCreateEx - Initializes a Memory Pool created by an Application with Application specified block sizes
  - CFE_ES_PoolCreateNoSem - Initializes a Memory Pool created by an Application without using a semaphore
  - CFE_ES_GetPoolBuf - Gets a buffer from a Memory Pool
  - CFE_ES_PutPoolBuf - Releases a buffer to a Memory Pool
  - CFE_ES_GetMemPoolStats - Extracts statistics maintained by the memory pool software
  - CFE_ES_GetPoolBufInfo - Gets info on a buffer previously allocated from a Memory Pool
- Performance Monitoring APIs
  - CFE_ES_PerfLogEntry - Marks the entry into a performance analyzer segment of code
  - CFE_ES_PerfLogExit - Marks the exit from a performance analyzer segment of code
  - CFE_ES_PerfLogAdd - Adds a mark into the Performance Log
- Generic Counter APIs
  - CFE_ES_RegisterGenCounter - Registers a generic counter
  - CFE_ES_DeleteGenCounter - Delete previously registered generic counter
  - CFE_ES_IncrementGenCounter - Increments the specified generic counter
  - CFE_ES_SetGenCount - Set the specified generic counter
  - CFE_ES_GetGenCount - Get the specified generic counter count
  - CFE_ES_GetGenCounterIDByName - Get the Id associated with a generic counter name
- Miscellaneous APIs
  - CFE_ES_CalculateCRC - Calculates a CRC on a block of memory
  - CFE_ES_WriteToSysLog - Writes a string to the System Log
  - CFE_ES_ProcessCoreException - Process an exception detected by the underlying OS/PSP

Events Services API

- Registration APIs
  - CFE_EVS_Register - Register an Application for using Event Services
  - CFE_EVS_Unregister - Unregister an Application from using Event Services
- Send Event APIs
  - CFE_EVS_SendEvent - Generate a Software Event
  - CFE_EVS_SendEventWithAppID - Generate a Software Event as though produced by the specified Application
  - CFE_EVS_SendTimedEvent - Generate a Software Event with a specified time tag
- Reset Event Filter APIs
  - CFE_EVS_ResetFilter - Resets the calling Application's Event Filter for a specified event
  - CFE_EVS_ResetAllFilters - Resets all of the calling Application's Event Filters

File Services API

- cFE File Header Management APIs
  - CFE_FS_ReadHeader - Read the contents of the Standard cFE File Header
  - CFE_FS_InitHeader - Initializes the contents of the Standard cFE File Header
  - CFE_FS_WriteHeader - Write the contents of the Standard cFE File Header
  - CFE_FS_SetTimestamp - Modify the Time Stamp field in the Standard cFE File Header
- Compressed File Management APIs
  - CFE_FS_IsGzFile - Determines if specified file is a gzip/compressed file
  - CFE_FS_Decompress - Decompresses the specified file to a destination file
  - CFE_FS_GetUncompressedFile - Decompresses the source file to a temporary file created in the temp dir
- Filename Utility APIs
  - CFE_FS_ExtractFilenameFromPath - Extracts the filename from a unix style path

Software Bus API

- Pipe Management APIs
  - CFE_SB_CreatePipe - Creates a new software bus pipe
  - CFE_SB_DeletePipe - Deletes a software bus pipe
  - CFE_SB_SetPipeOpts - Set options on a pipe
  - CFE_SB_GetPipeOpts - Get options on a pipe
- Message Subscription Control APIs
  - CFE_SB_Subscribe - Subscribe to a message on the software bus with default parameters
  - CFE_SB_SubscribeEx - Subscribe to a message on the software bus
  - CFE_SB_SubscribeLocal - Subscribe to a message while keeping the request on the current CPU
  - CFE_SB_Unsubscribe - Remove a subscription to a message on the software bus

Table Services API

- Registration APIs
  - CFE_TBL_Register - Register a table with cFE to obtain Table Management Services
  - CFE_TBL_Share - Obtain access to a table registered by another Application
  - CFE_TBL_Unregister - Unregister a previously registered table and free resources
- Manage Table Content APIs
  - CFE_TBL_Load - Load a specified table with data
  - CFE_TBL_Update - Update the contents of a table with any pending data
  - CFE_TBL_Validate - Validate the contents of a table
  - CFE_TBL_Manage - Perform standard routine operations to maintain a table
  - CFE_TBL_DumpToBuffer - Copies contents of a Dump Only Table to a shared buffer
  - CFE_TBL_Modified - Notifies Table Services that table contents have been modified by Application
- Access Table Content APIs
  - CFE_TBL_GetAddress - Obtain the current address of the contents of a table
  - CFE_TBL_GetAddresses - Obtain the current addresses of multiple tables
  - CFE_TBL_ReleaseAddress - Release the pointer to the contents of a table
  - CFE_TBL_ReleaseAddresses - Release the pointers to the contents of multiple tables
- Get Table Information APIs
  - CFE_TBL_GetStatus - Obtain current status of pending actions for a table
  - CFE_TBL_GetInfo - Obtain characteristics/information about a specific table
  - CFE_TBL_NotifyByMessage - Request notification via message when table requires management

Time Services API

- Get Current Time APIs
  - CFE_TIME_GetTime - Get the current spacecraft time
  - CFE_TIME_GetTAI - Get the current TAI time
  - CFE_TIME_GetUTC - Get the current UTC time
  - CFE_TIME_GetMET - Get the current Mission Elapsed Time
  - CFE_TIME_GetMETseconds - Get the current seconds count of the Mission Elapsed Time
  - CFE_TIME_GetMETsubsecs - Get the current subseconds count of the Mission Elapsed Time
- Get Time Information APIs
  - CFE_TIME_GetSTCF - Get the current value of the Spacecraft Time Correction Factor
  - CFE_TIME_GetLeapSeconds - Get the current value of the leap seconds counter
  - CFE_TIME_GetClockState - Get the state of the spacecraft clock
  - CFE_TIME_GetClockInfo - Get information about the spacecraft clock
- Time Arithmetic APIs
  - CFE_TIME_Add - Adds two time values

9 cFE Mission Configuration Parameters

Global CFE_MISSION_CMD_MID_BASE1

cFE Message ID Base Numbers

Global CFE_MISSION_ES_HK_TLM_MSG

cFE Portable Message Numbers for Telemetry

Global CFE_MISSION_EVS_CMD_MSG

cFE Portable Message Numbers for Commands

Global CFE_MISSION_MAX_API_LEN

cFE Maximum length for API names within data exchange structures

Global CFE_MISSION_MAX_FILE_LEN

cFE Maximum length for filenames within data exchange structures

Global CFE_MISSION_MAX_PATH_LEN

cFE Maximum length for pathnames within data exchange structures

Global CFE_MISSION_SB_PACKET_TIME_FORMAT

Packet Timestamp Format Selection

Global CFE_MISSION_SPACECRAFT_ID
Spacecraft ID

Global CFE_MISSION_TIME_DATA_CMD_MSG
cFE Portable Message Numbers for Global Messages

Global MESSAGE_FORMAT_IS_CCSDS
cFE SB message format

10 Data Structure Index

10.1 Data Structures

Here are the data structures with brief descriptions:

11 File Index

11.1 File List

Here is a list of all files with brief descriptions:

12 Data Structure Documentation

12.1 BD Struct Reference

`#include <cfe_esmempool.h>`

Data Fields

- uint16 CheckBits
- uint16 Allocated
- uint32 Size
- BD_t * Next

12.1.1 Detailed Description

Definition at line 42 of file cfe_esmempool.h.

12.1.2 Field Documentation

12.1.2.1 Allocated

`uint16 BD::Allocated`

Definition at line 45 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

12.1.2.2 CheckBits

```
uint16 BD::CheckBits
```

Definition at line 44 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

12.1.2.3 Next

```
BD_t* BD::Next
```

Definition at line 47 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), and CFE_ES_PutPoolBuf().

12.1.2.4 Size

```
uint32 BD::Size
```

Definition at line 46 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_esmempool.h

12.2 BlockSizeDesc_t Struct Reference

```
#include <cfe_esmempool.h>
```

Data Fields

- BD_t * Top
- uint32 NumCreated
- uint32 NumFree
- uint32 MaxSize

12.2.1 Detailed Description

Definition at line 50 of file cfe_esmempool.h.

12.2.2 Field Documentation

12.2.2.1 MaxSize

```
uint32 BlockSizeDesc_t::MaxSize
```

Definition at line 55 of file cfe_esmempool.h.

Referenced by CFE_ES_GetBlockSize(), CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_Pool↩CreateEx(), and CFE_ES_PutPoolBuf().

12.2.2.2 NumCreated

```
uint32 BlockSizeDesc_t::NumCreated
```

Definition at line 53 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), and CFE_ES_PoolCreateEx().

12.2.2.3 NumFree

```
uint32 BlockSizeDesc_t::NumFree
```

Definition at line 54 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_PoolCreateEx(), and CFE_ES_Put↩PoolBuf().

12.2.2.4 Top

```
BD_t* BlockSizeDesc_t::Top
```

Definition at line 52 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_PoolCreateEx(), and CFE_ES_PutPoolBuf().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_esmempool.h

12.3 CCSDS_APIDQHdr_t Struct Reference

CCSDS Primary with APID Qualifier Header Type Definition.

`#include <ccsds.h>`

Data Fields

- CCSDS_PriHdr_t Pri
  CCSDS Primary Header CCSDS_PriHdr_t.
- CCSDS_APIDqualifiers_t ApidQ
  CCSDS APID Qualifier Secondary Header CCSDS_APIDqualifiers_t.

12.3.1 Detailed Description

Definition at line 161 of file ccsds.h.

12.3.2 Field Documentation

12.3.2.1 ApidQ

`CCSDS_APIDqualifiers_t CCSDS_APIDQHdr_t::ApidQ`

Definition at line 163 of file ccsds.h.

12.3.2.2 Pri

`CCSDS_PriHdr_t CCSDS_APIDQHdr_t::Pri`

Definition at line 162 of file ccsds.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.4 CCSDS_APIDqualifiers_t Struct Reference

`#include <ccsds.h>`

Data Fields

- uint8 APIDQSubsystem [2]
- uint8 APIDQSystemId [2]

12.4.1 Detailed Description

Definition at line 143 of file ccsds.h.

12.4.2 Field Documentation

12.4.2.1 APIDQSubsystem

```
uint8 CCSDS_APIDqualifiers_t::APIDQSubsystem[2]
```

Definition at line 145 of file ccsds.h.

12.4.2.2 APIDQSystemId

```
uint8 CCSDS_APIDqualifiers_t::APIDQSystemId[2]
```

Definition at line 153 of file ccsds.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.5 CCSDS_CmdSecHdr_t Struct Reference

```
#include <ccsds.h>
```

Data Fields

- uint16 Command

12.5.1 Detailed Description

Definition at line 108 of file ccsds.h.

12.5.2 Field Documentation

12.5.2.1 Command

```
uint16 CCSDS_CmdSecHdr_t::Command
```

Definition at line 110 of file ccsds.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.6 CCSDS_CommandPacket_t Struct Reference

```
#include <ccsds.h>
```

Data Fields

- CCSDS_SpacePacket_t SpacePacket
  Standard Header on all packets.
- CCSDS_CmdSecHdr_t Sec

12.6.1 Detailed Description

Definition at line 188 of file ccsds.h.

12.6.2 Field Documentation

12.6.2.1 Sec

```
CCSDS_CmdSecHdr_t CCSDS_CommandPacket_t::Sec
```

Definition at line 191 of file ccsds.h.

Referenced by CCSDS_LoadCheckSum(), CFE_SB_GetChecksum(), CFE_SB_GetCmdCode(), and CFE_SB_Set↩CmdCode().

12.6.2.2 SpacePacket

```
CCSDS_SpacePacket_t CCSDS_CommandPacket_t::SpacePacket
```

Definition at line 190 of file ccsds.h.

Referenced by CCSDS_ComputeCheckSum().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.7 CCSDS_PriHdr_t Struct Reference

```
#include <ccsds.h>
```

Data Fields

- uint8 StreamId [2]
- uint8 Sequence [2]
- uint8 Length [2]

12.7.1 Detailed Description

Definition at line 86 of file ccsds.h.

12.7.2 Field Documentation

12.7.2.1 Length

```
uint8 CCSDS_PriHdr_t::Length[2]
```

Definition at line 100 of file ccsds.h.

12.7.2.2 Sequence

```
uint8 CCSDS_PriHdr_t::Sequence[2]
```

Definition at line 95 of file ccsds.h.

12.7.2.3 StreamId

```
uint8 CCSDS_PriHdr_t::StreamId[2]
```

Definition at line 88 of file ccsds.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.8 CCSDS_SpacePacket_t Struct Reference

```
#include <ccsds.h>
```

Data Fields

- CCSDS_PriHdr_t Hdr

12.8.1 Detailed Description

Definition at line 166 of file ccsds.h.

12.8.2 Field Documentation

12.8.2.1 Hdr

```
CCSDS_PriHdr_t CCSDS_SpacePacket_t::Hdr
```

Complete "version 1" (standard) header

Definition at line 171 of file ccsds.h.

Referenced by CCSDS_ComputeCheckSum(), and CFE_SB_SetMsgId().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.9 CCSDS_TelemetryPacket_t Struct Reference

```
#include <ccsds.h>
```

Data Fields

- CCSDS_SpacePacket_t SpacePacket
  Standard Header on all packets.
- CCSDS_TlmSecHdr_t Sec

12.9.1 Detailed Description

Definition at line 196 of file ccsds.h.

12.9.2 Field Documentation

12.9.2.1 Sec

`CCSDS_TlmSecHdr_t CCSDS_TelemetryPacket_t::Sec`

Definition at line 199 of file ccsds.h.

Referenced by CFE_SB_GetMsgTime(), and CFE_SB_SetMsgTime().

12.9.2.2 SpacePacket

`CCSDS_SpacePacket_t CCSDS_TelemetryPacket_t::SpacePacket`

Definition at line 198 of file ccsds.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.10 CCSDS_TlmSecHdr_t Struct Reference

`#include <ccsds.h>`

Data Fields

- uint8 Time [CCSDS_TIME_SIZE]

12.10.1 Detailed Description

Definition at line 120 of file ccsds.h.

12.10.2 Field Documentation

12.10.2.1 Time

```
uint8 CCSDS_TlmSecHdr_t::Time[CCSDS_TIME_SIZE]
```

Definition at line 122 of file ccsds.h.

Referenced by CFE_SB_GetMsgTime(), and CFE_SB_SetMsgTime().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/ccsds.h

12.11 CFE_ES_AppInfo_t Struct Reference

```
#include <cfe_es.h>
```

Data Fields

- uint32 AppId
  Application ID for this Application.
- uint32 Type
  The type of App: CORE or EXTERNAL.
- char Name [OS_MAX_API_NAME]
  The Registered Name of the Application.
- char EntryPoint [OS_MAX_API_NAME]
  The Entry Point label for the Application.
- char FileName [OS_MAX_PATH_LEN]
  The Filename of the file containing the Application.
- uint32 StackSize
  The Stack Size of the Application.
- uint32 ModuleId
  The ID of the Loadable Module for the Application.
- uint32 AddressesAreValid
  Indicates that the Code, Data, and BSS addresses/sizes are valid.
- uint32 CodeAddress
  The Address of the Application Code Segment.
- uint32 CodeSize

The Code Size of the Application.

- uint32 DataAddress

  The Address of the Application Data Segment.
- uint32 DataSize

  The Data Size of the Application.
- uint32 BSSAddress

  The Address of the Application BSS Segment.
- uint32 BSSSize

  The BSS Size of the Application.
- uint32 StartAddress

  The Start Address of the Application.
- uint16 ExceptionAction

  What should occur if Application has an exception (Restart Application OR Restart Processor)
- uint16 Priority

  The Priority of the Application.
- uint32 MainTaskId

  The Application's Main Task ID.
- uint32 ExecutionCounter

  The Application's Main Task Execution Counter.
- char MainTaskName [OS_MAX_API_NAME]

  The Application's Main Task ID.
- uint32 NumOfChildTasks

  Number of Child tasks for an App.

12.11.1 Detailed Description

Definition at line 205 of file cfe_es.h.

12.11.2 Field Documentation

12.11.2.1 AddressesAreValid

```
uint32 CFE_ES_AppInfo_t::AddressesAreValid
```

Telemetry Mnemonic(s) **`$sc_$cpu_ES_AddrsValid`**

Definition at line 223 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.2 AppId

uint32 CFE_ES_AppInfo_t::AppId

Telemetry Mnemonic(s) **$sc_$cpu_ES_AppID**

Definition at line 207 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.3 BSSAddress

uint32 CFE_ES_AppInfo_t::BSSAddress

Telemetry Mnemonic(s) **$sc_$cpu_ES_BSSAddress**

Definition at line 233 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.4 BSSSize

uint32 CFE_ES_AppInfo_t::BSSSize

Telemetry Mnemonic(s) **$sc_$cpu_ES_BSSSize**

Definition at line 235 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.5 CodeAddress

uint32 CFE_ES_AppInfo_t::CodeAddress

Telemetry Mnemonic(s) **$sc_$cpu_ES_CodeAddress**

Definition at line 225 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.6 CodeSize

```
uint32 CFE_ES_AppInfo_t::CodeSize
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CodeSize**

Definition at line 227 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.7 DataAddress

```
uint32 CFE_ES_AppInfo_t::DataAddress
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_DataAddress**

Definition at line 229 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.8 DataSize

```
uint32 CFE_ES_AppInfo_t::DataSize
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_DataSize**

Definition at line 231 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.9 EntryPoint

```
char CFE_ES_AppInfo_t::EntryPoint[OS_MAX_API_NAME]
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_AppEntryPt[OS_MAX_API_NAME]**

Definition at line 214 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.10 ExceptionAction

```
uint16 CFE_ES_AppInfo_t::ExceptionAction
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ExceptnActn**

Definition at line 239 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.11 ExecutionCounter

```
uint32 CFE_ES_AppInfo_t::ExecutionCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ExecutionCtr**

Definition at line 246 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.12 FileName

```
char CFE_ES_AppInfo_t::FileName[OS_MAX_PATH_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_AppFilename[OS_MAX_PATH_LEN]**

Definition at line 216 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.13 MainTaskId

```
uint32 CFE_ES_AppInfo_t::MainTaskId
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_MainTaskId**

Definition at line 244 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.14 MainTaskName

```
char CFE_ES_AppInfo_t::MainTaskName[OS_MAX_API_NAME]
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_MainTaskName[OS_MAX_API_NAME]**

Definition at line 248 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.15 ModuleId

```
uint32 CFE_ES_AppInfo_t::ModuleId
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ModuleID**

Definition at line 221 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.16 Name

```
char CFE_ES_AppInfo_t::Name[OS_MAX_API_NAME]
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_AppName[OS_MAX_API_NAME]**

Definition at line 212 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.17 NumOfChildTasks

```
uint32 CFE_ES_AppInfo_t::NumOfChildTasks
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ChildTasks**

Definition at line 250 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.18 Priority

```
uint16 CFE_ES_AppInfo_t::Priority
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_Priority**

Definition at line 242 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.19 StackSize

```
uint32 CFE_ES_AppInfo_t::StackSize
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_StackSize**

Definition at line 219 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.20 StartAddress

```
uint32 CFE_ES_AppInfo_t::StartAddress
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_StartAddr**

Definition at line 237 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.11.2.21 Type

```
uint32 CFE_ES_AppInfo_t::Type
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_AppType**

Definition at line 209 of file cfe_es.h.

Referenced by CFE_ES_GetAppInfoInternal().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es.h

12.12 CFE_ES_AppNameCmd_Payload_t Struct Reference

Command Structure for Commands requiring just an Application Name.

`#include <cfe_es_msg.h>`

Data Fields

- char Application [CFE_MISSION_MAX_API_LEN]
  ASCII text string containing Application Name.

12.12.1 Detailed Description

For command details, see CFE_ES_STOP_APP_CC, CFE_ES_RESTART_APP_CC, CFE_ES_QUERY_ONE_CC

Definition at line 1258 of file cfe_es_msg.h.

12.12.2 Field Documentation

12.12.2.1 Application

`char CFE_ES_AppNameCmd_Payload_t::Application[CFE_MISSION_MAX_API_LEN]`

Definition at line 1260 of file cfe_es_msg.h.

Referenced by CFE_ES_QueryOneCmd(), CFE_ES_RestartAppCmd(), and CFE_ES_StopAppCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.13 CFE_ES_AppNameCmd_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_AppNameCmd_Payload_t Payload

12.13.1 Detailed Description

Definition at line 1263 of file cfe_es_msg.h.

12.13.2 Field Documentation

12.13.2.1 CmdHeader

```
uint8 CFE_ES_AppNameCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1265 of file cfe_es_msg.h.

12.13.2.2 Payload

```
CFE_ES_AppNameCmd_Payload_t CFE_ES_AppNameCmd_t::Payload
```

Definition at line 1266 of file cfe_es_msg.h.

Referenced by CFE_ES_QueryOneCmd(), CFE_ES_RestartAppCmd(), and CFE_ES_StopAppCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.14 CFE_ES_AppRecord_t Struct Reference

```
#include <cfe_es_apps.h>
```

Data Fields

- CFE_ES_AppState_Enum_t AppState
- uint32 Type
- CFE_ES_AppStartParams_t StartParams
- CFE_ES_ControlReq_t ControlReq
- CFE_ES_MainTaskInfo_t TaskInfo

12.14.1 Detailed Description

Definition at line 102 of file cfe_es_apps.h.

12.14.2 Field Documentation

12.14.2.1 AppState

CFE_ES_AppState_Enum_t CFE_ES_AppRecord_t::AppState

Definition at line 104 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfo(), CFE_ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_ES_ListApplications(), CFE_ES_Main(), CFE_ES_MainTaskSyncDelay(), CFE_ES_QueryAllCmd(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_ScanAppTable(), and CFE_ES_SetAppState().

12.14.2.2 ControlReq

CFE_ES_ControlReq_t CFE_ES_AppRecord_t::ControlReq

Definition at line 107 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_ES_ProcessControlRequest(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), and CFE_ES_ScanAppTable().

12.14.2.3 StartParams

CFE_ES_AppStartParams_t CFE_ES_AppRecord_t::StartParams

Definition at line 106 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_ES_ListApplications(), CFE_ES_ProcessControlRequest(), CFE_ES_ProcessCoreException(), CFE_ES_ReloadApp(), and CFE_ES_RestartApp().

12.14.2.4 TaskInfo

CFE_ES_MainTaskInfo_t CFE_ES_AppRecord_t::TaskInfo

Definition at line 108 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), CFE_ES_GetAppInfoInternal(), and CFE_ES_RunLoop().

12.14.2.5 Type

```
uint32 CFE_ES_AppRecord_t::Type
```

Definition at line 105 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_ES_GetAppInfoInternal(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_ScanAppTable(), and CFE_ES_WaitForSystemState().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.h

12.15 CFE_ES_AppReloadCmd_Payload_t Struct Reference

Reload Application Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- char Application [CFE_MISSION_MAX_API_LEN]
  ASCII text string containing Application Name.
- char AppFileName [CFE_MISSION_MAX_PATH_LEN]
  Full path and filename of Application's executable image.

12.15.1 Detailed Description

For command details, see CFE_ES_RELOAD_APP_CC

Definition at line 1284 of file cfe_es_msg.h.

12.15.2 Field Documentation

12.15.2.1 AppFileName

```
char CFE_ES_AppReloadCmd_Payload_t::AppFileName[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 1287 of file cfe_es_msg.h.

Referenced by CFE_ES_ReloadAppCmd().

12.15.2.2 Application

```
char CFE_ES_AppReloadCmd_Payload_t::Application[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1286 of file cfe_es_msg.h.

Referenced by CFE_ES_ReloadAppCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.16 CFE_ES_AppStartParams_t Struct Reference

```
#include <cfe_es_apps.h>
```

Data Fields

- char Name [OS_MAX_API_NAME]
- char EntryPoint [OS_MAX_API_NAME]
- char FileName [OS_MAX_PATH_LEN]
- uint32 StackSize
- cpuaddr StartAddress
- uint32 ModuleId
- uint16 ExceptionAction
- uint16 Priority

12.16.1 Detailed Description

Definition at line 71 of file cfe_es_apps.h.

12.16.2 Field Documentation

12.16.2.1 EntryPoint

```
char CFE_ES_AppStartParams_t::EntryPoint[OS_MAX_API_NAME]
```

Definition at line 74 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_GetAppInfoInternal(), and CFE_ES_ProcessControlRequest().

12.16.2.2 ExceptionAction

```
uint16 CFE_ES_AppStartParams_t::ExceptionAction
```

Definition at line 81 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateObjects(), CFE_ES_GetAppInfoInternal(), CFE_ES_Process↩ ControlRequest(), and CFE_ES_ProcessCoreException().

12.16.2.3 FileName

```
char CFE_ES_AppStartParams_t::FileName[OS_MAX_PATH_LEN]
```

Definition at line 75 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_GetAppInfoInternal(), CFE_ES_ProcessControlRequest(), and CFE↩ _ES_ReloadApp().

12.16.2.4 ModuleId

```
uint32 CFE_ES_AppStartParams_t::ModuleId
```

Definition at line 79 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), and CFE_ES_GetAppInfoInternal().

12.16.2.5 Name

```
char CFE_ES_AppStartParams_t::Name[OS_MAX_API_NAME]
```

Definition at line 73 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_↩ ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_E↩ S_ListApplications(), CFE_ES_ProcessControlRequest(), CFE_ES_ReloadApp(), and CFE_ES_RestartApp().

12.16.2.6 Priority

```
uint16 CFE_ES_AppStartParams_t::Priority
```

Definition at line 82 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateObjects(), CFE_ES_GetAppInfoInternal(), and CFE_ES_↩ ProcessControlRequest().

12.16.2.7 StackSize

```
uint32 CFE_ES_AppStartParams_t::StackSize
```

Definition at line 77 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateObjects(), CFE_ES_GetAppInfoInternal(), and CFE_ES_↩ ProcessControlRequest().

12.16.2.8 StartAddress

```
cpuaddr CFE_ES_AppStartParams_t::StartAddress
```

Definition at line 78 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateObjects(), and CFE_ES_GetAppInfoInternal().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.h

12.17 CFE_ES_BlockStats_t Struct Reference

```
#include <cfe_es.h>
```

Data Fields

- uint32 BlockSize

  Number of bytes in each of these blocks.
- uint32 NumCreated

  Number of Memory Blocks of this size created.
- uint32 NumFree

  Number of Memory Blocks of this size that are free.

12.17.1 Detailed Description

Definition at line 271 of file cfe_es.h.

12.17.2 Field Documentation

12.17.2.1 BlockSize

```
uint32 CFE_ES_BlockStats_t::BlockSize
```

Definition at line 273 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

12.17.2.2 NumCreated

```
uint32 CFE_ES_BlockStats_t::NumCreated
```

Definition at line 274 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

12.17.2.3 NumFree

```
uint32 CFE_ES_BlockStats_t::NumFree
```

Definition at line 275 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es.h

12.18 CFE_ES_CDS_RegRec_t Struct Reference

```
#include <cfe_es_cds.h>
```

Data Fields

- char Name [CFE_ES_CDS_MAX_FULL_NAME_LEN]
- CFE_ES_CDSBlockHandle_t MemHandle
- uint32 Size

  Size, in bytes, of the CDS memory block.
- bool Taken

  Flag that indicates whether the registry record is in use.
- bool Table

  Flag that indicates whether CDS contains a Critical Table.

12.18.1 Detailed Description

Definition at line 65 of file cfe_es_cds.h.

12.18.2 Field Documentation

12.18.2.1 MemHandle

```
CFE_ES_CDSBlockHandle_t CFE_ES_CDS_RegRec_t::MemHandle
```

Definition at line 68 of file cfe_es_cds.h.

Referenced by CFE_ES_CopyToCDS(), CFE_ES_DeleteCDS(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_InitC↩DSRegistry(), CFE_ES_RegisterCDSEx(), and CFE_ES_RestoreFromCDS().

12.18.2.2 Name

```
char CFE_ES_CDS_RegRec_t::Name[CFE_ES_CDS_MAX_FULL_NAME_LEN]
```

Definition at line 67 of file cfe_es_cds.h.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_FindCDSInRegistry(), CFE_E↩S_InitCDSRegistry(), and CFE_ES_RegisterCDSEx().

12.18.2.3 Size

```
uint32 CFE_ES_CDS_RegRec_t::Size
```

Definition at line 69 of file cfe_es_cds.h.

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_InitCDSRegistry(), and CFE_ES_RegisterCDSEx().

12.18.2.4 Table

```
bool CFE_ES_CDS_RegRec_t::Table
```

Definition at line 71 of file cfe_es_cds.h.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_InitCDSRegistry(), and CFE_↩ES_RegisterCDSEx().

12.18.2.5 Taken

```
bool CFE_ES_CDS_RegRec_t::Taken
```

Definition at line 70 of file cfe_es_cds.h.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_FindCDSInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), CFE_ES_InitCDSRegistry(), and CFE_ES_RegisterCDSEx().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_cds.h

12.19 CFE_ES_CDSBlockDesc_t Struct Reference

```
#include <cfe_es_cds_mempool.h>
```

Data Fields

- uint16 CheckBits
- uint16 AllocatedFlag
- uint32 SizeUsed
- uint32 ActualSize
- uint32 CRC
- uint32 Next

12.19.1 Detailed Description

Definition at line 57 of file cfe_es_cds_mempool.h.

12.19.2 Field Documentation

12.19.2.1 ActualSize

```
uint32 CFE_ES_CDSBlockDesc_t::ActualSize
```

Definition at line 62 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

12.19.2.2 AllocatedFlag

```
uint16 CFE_ES_CDSBlockDesc_t::AllocatedFlag
```

Definition at line 60 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCD↩SBlock(), and CFE_ES_RebuildCDSPool().

12.19.2.3 CheckBits

```
uint16 CFE_ES_CDSBlockDesc_t::CheckBits
```

Definition at line 59 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCD↩SBlock(), and CFE_ES_RebuildCDSPool().

12.19.2.4 CRC

```
uint32 CFE_ES_CDSBlockDesc_t::CRC
```

Definition at line 63 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), and CFE_ES_GetCDSBlock().

12.19.2.5 Next

```
uint32 CFE_ES_CDSBlockDesc_t::Next
```

Definition at line 64 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_GetCDSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

12.19.2.6 SizeUsed

```
uint32 CFE_ES_CDSBlockDesc_t::SizeUsed
```

Definition at line 61 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), and CFE_ES_↩RebuildCDSPool().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_cds_mempool.h

12.20 CFE_ES_CDSBlockSizeDesc_t Struct Reference

`#include <cfe_es_cds_mempool.h>`

Data Fields

- uint32 Top
- uint32 NumCreated
- uint32 MaxSize

12.20.1 Detailed Description

Definition at line 67 of file cfe_es_cds_mempool.h.

12.20.2 Field Documentation

12.20.2.1 MaxSize

`uint32 CFE_ES_CDSBlockSizeDesc_t::MaxSize`

Definition at line 71 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSGetBinIndex(), CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), and CFE_ES_↩ RebuildCDSPool().

12.20.2.2 NumCreated

`uint32 CFE_ES_CDSBlockSizeDesc_t::NumCreated`

Definition at line 70 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), and CFE_ES_RebuildCDSPool().

12.20.2.3 Top

`uint32 CFE_ES_CDSBlockSizeDesc_t::Top`

Definition at line 69 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_↩ RebuildCDSPool().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_cds_mempool.h

12.21 CFE_ES_CDSPool_t Struct Reference

`#include <cfe_es_cds_mempool.h>`

Data Fields

- uint32 Start
- uint32 Size
- uint32 End
- uint32 Current
- int32 SizeIndex
- uint16 CheckErrCntr
- uint16 RequestCntr
- uint32 MutexId
- uint32 MinBlockSize
- CFE_ES_CDSBlockSizeDesc_t SizeDesc [CFE_ES_CDS_NUM_BLOCK_SIZES]

12.21.1 Detailed Description

Definition at line 76 of file cfe_es_cds_mempool.h.

12.21.2 Field Documentation

12.21.2.1 CheckErrCntr

`uint16 CFE_ES_CDSPool_t::CheckErrCntr`

Definition at line 82 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CreateCDSPool(), CFE_ES_PutC↩DSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.2 Current

`uint32 CFE_ES_CDSPool_t::Current`

Definition at line 80 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.3 End

```
uint32 CFE_ES_CDSPool_t::End
```

Definition at line 79 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CreateCDSPool(), CFE_ES_GetC↩ DSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.4 MinBlockSize

```
uint32 CFE_ES_CDSPool_t::MinBlockSize
```

Definition at line 85 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CDSReqdMinSize(), CFE_ES_↩ CreateCDSPool(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.5 MutexId

```
uint32 CFE_ES_CDSPool_t::MutexId
```

Definition at line 84 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CreateCDSPool(), CFE_ES_GetC↩ DSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.6 RequestCntr

```
uint16 CFE_ES_CDSPool_t::RequestCntr
```

Definition at line 83 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.7 Size

```
uint32 CFE_ES_CDSPool_t::Size
```

Definition at line 78 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), and CFE_ES_RebuildCDSPool().

12.21.2.8 SizeDesc

```
CFE_ES_CDSBlockSizeDesc_t CFE_ES_CDSPool_t::SizeDesc[CFE_ES_CDS_NUM_BLOCK_SIZES]
```

Definition at line 86 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSGetBinIndex(), CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), CFE_ES_PutC↩ DSBlock(), and CFE_ES_RebuildCDSPool().

12.21.2.9 SizeIndex

```
int32 CFE_ES_CDSPool_t::SizeIndex
```

Definition at line 81 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), and CFE_ES_RebuildCDSPool().

12.21.2.10 Start

```
uint32 CFE_ES_CDSPool_t::Start
```

Definition at line 77 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CreateCDSPool(), and CFE_ES_RebuildCDSPool().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_cds_mempool.h

12.22 CFE_ES_CDSRegDumpRec_t Struct Reference

```
#include <cfe_es.h>
```

Data Fields

- CFE_ES_CDSHandle_t Handle
  Handle of CDS.
- uint32 Size
  Size, in bytes, of the CDS memory block.
- bool Table
  Flag that indicates whether CDS contains a Critical Table.
- char Name [CFE_ES_CDS_MAX_FULL_NAME_LEN]
  Processor Unique Name of CDS.
- uint8 ByteAlignSpare1
  Spare byte to insure structure size is multiple of 4 bytes.

12.22.1 Detailed Description

Definition at line 297 of file cfe_es.h.

12.22.2 Field Documentation

12.22.2.1 ByteAlignSpare1

```
uint8 CFE_ES_CDSRegDumpRec_t::ByteAlignSpare1
```

Definition at line 303 of file cfe_es.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

12.22.2.2 Handle

```
CFE_ES_CDSHandle_t CFE_ES_CDSRegDumpRec_t::Handle
```

Definition at line 299 of file cfe_es.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

12.22.2.3 Name

```
char CFE_ES_CDSRegDumpRec_t::Name[CFE_ES_CDS_MAX_FULL_NAME_LEN]
```

Definition at line 302 of file cfe_es.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

12.22.2.4 Size

```
uint32 CFE_ES_CDSRegDumpRec_t::Size
```

Definition at line 300 of file cfe_es.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

12.22.2.5 Table

```
bool CFE_ES_CDSRegDumpRec_t::Table
```

Definition at line 301 of file cfe_es.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es.h

12.23 CFE_ES_CDSVariables_t Struct Reference

```
#include <cfe_es_cds.h>
```

Data Fields

- uint32 RegistryMutex

  Mutex that controls access to CDS Registry.
- uint32 CDSSize

  Total size of the CDS as reported by BSP.
- uint32 MemPoolSize
- uint32 MaxNumRegEntries

  Maximum number of Registry entries.
- CFE_ES_CDS_RegRec_t Registry [CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES]

  CDS Registry (Local Copy)
- char ValidityField [8]

12.23.1 Detailed Description

Definition at line 74 of file cfe_es_cds.h.

12.23.2 Field Documentation

12.23.2.1 CDSSize

```
uint32 CFE_ES_CDSVariables_t::CDSSize
```

Definition at line 77 of file cfe_es_cds.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_RebuildCDS(), and CFE_ES_ValidateCDS().

12.23.2.2 MaxNumRegEntries

```
uint32 CFE_ES_CDSVariables_t::MaxNumRegEntries
```

Definition at line 79 of file cfe_es_cds.h.

Referenced by CFE_ES_FindCDSInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), CFE_ES_InitCDSRegistry(), and CFE_ES_RebuildCDS().

12.23.2.3 MemPoolSize

```
uint32 CFE_ES_CDSVariables_t::MemPoolSize
```

Definition at line 78 of file cfe_es_cds.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_InitializeCDS(), CFE_ES_RebuildCDS(), and CFE_ES_Register↩ CDS().

12.23.2.4 Registry

```
CFE_ES_CDS_RegRec_t CFE_ES_CDSVariables_t::Registry[CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES]
```

Definition at line 80 of file cfe_es_cds.h.

Referenced by CFE_ES_CopyToCDS(), CFE_ES_DeleteCDS(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_Find↩ CDSInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), CFE_ES_InitCDSRegistry(), CFE_ES_RebuildCDS(), CF↩ E_ES_RegisterCDSEx(), CFE_ES_RestoreFromCDS(), and CFE_ES_UpdateCDSRegistry().

12.23.2.5 RegistryMutex

```
uint32 CFE_ES_CDSVariables_t::RegistryMutex
```

Definition at line 76 of file cfe_es_cds.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_LockCDSRegistry(), and CFE_ES_UnlockCDSRegistry().

12.23.2.6 ValidityField

```
char CFE_ES_CDSVariables_t::ValidityField[8]
```

Definition at line 81 of file cfe_es_cds.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_↩ InitializeCDS(), CFE_ES_PutCDSBlock(), CFE_ES_RebuildCDS(), and CFE_ES_ValidateCDS().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_cds.h

12.24 CFE_ES_CleanupState_t Struct Reference

Data Fields

- uint32 ErrorFlag
- uint32 FoundObjects
- uint32 PrevFoundObjects
- uint32 DeletedObjects
- int32 OverallStatus

12.24.1 Detailed Description

Definition at line 1306 of file cfe_es_apps.c.

12.24.2 Field Documentation

12.24.2.1 DeletedObjects

```
uint32 CFE_ES_CleanupState_t::DeletedObjects
```

Definition at line 1311 of file cfe_es_apps.c.

Referenced by CFE_ES_CleanupObjectCallback(), and CFE_ES_CleanupTaskResources().

12.24.2.2 ErrorFlag

```
uint32 CFE_ES_CleanupState_t::ErrorFlag
```

Definition at line 1308 of file cfe_es_apps.c.

Referenced by CFE_ES_CleanupTaskResources().

12.24.2.3 FoundObjects

```
uint32 CFE_ES_CleanupState_t::FoundObjects
```

Definition at line 1309 of file cfe_es_apps.c.

Referenced by CFE_ES_CleanupObjectCallback(), and CFE_ES_CleanupTaskResources().

12.24.2.4 OverallStatus

```
int32 CFE_ES_CleanupState_t::OverallStatus
```

Definition at line 1312 of file cfe_es_apps.c.

Referenced by CFE_ES_CleanupObjectCallback(), and CFE_ES_CleanupTaskResources().

12.24.2.5 PrevFoundObjects

```
uint32 CFE_ES_CleanupState_t::PrevFoundObjects
```

Definition at line 1310 of file cfe_es_apps.c.

Referenced by CFE_ES_CleanupTaskResources().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.c

12.25 CFE_ES_ControlReq_t Struct Reference

```
#include <cfe_es_apps.h>
```

Data Fields

- uint32 AppControlRequest
- int32 AppTimer

12.25.1 Detailed Description

Definition at line 59 of file cfe_es_apps.h.

12.25.2 Field Documentation

12.25.2.1 AppControlRequest

```
uint32 CFE_ES_ControlReq_t::AppControlRequest
```

Definition at line 61 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_ES_ProcessControlRequest(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), and CFE_ES_RunLoop().

12.25.2.2 AppTimer

```
int32 CFE_ES_ControlReq_t::AppTimer
```

Definition at line 62 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_DeleteApp(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), and CFE_ES_ScanAppTable().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.h

12.26 CFE_ES_DebugVariables_t Struct Reference

```
#include <cfe_es_erlog_typedef.h>
```

Data Fields

- uint32 DebugFlag
- uint32 WatchdogWriteFlag
- uint32 PrintfEnabledFlag
- uint32 LastAppId

12.26.1 Detailed Description

Definition at line 40 of file cfe_es_erlog_typedef.h.

12.26.2 Field Documentation

12.26.2.1 DebugFlag

```
uint32 CFE_ES_DebugVariables_t::DebugFlag
```

Definition at line 42 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_SetupResetVariables().

12.26.2.2 LastAppId

```
uint32 CFE_ES_DebugVariables_t::LastAppId
```

Definition at line 45 of file cfe_es_erlog_typedef.h.

12.26.2.3 PrintfEnabledFlag

```
uint32 CFE_ES_DebugVariables_t::PrintfEnabledFlag
```

Definition at line 44 of file cfe_es_erlog_typedef.h.

12.26.2.4 WatchdogWriteFlag

```
uint32 CFE_ES_DebugVariables_t::WatchdogWriteFlag
```

Definition at line 43 of file cfe_es_erlog_typedef.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_erlog_typedef.h

12.27 CFE_ES_DeleteCDS_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_DeleteCDSCmd_Payload_t Payload

12.27.1 Detailed Description

Definition at line 1327 of file cfe_es_msg.h.

12.27.2 Field Documentation

12.27.2.1 CmdHeader

```
uint8 CFE_ES_DeleteCDS_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1329 of file cfe_es_msg.h.

12.27.2.2 Payload

```
CFE_ES_DeleteCDSCmd_Payload_t CFE_ES_DeleteCDS_t::Payload
```

Definition at line 1330 of file cfe_es_msg.h.

Referenced by CFE_ES_DeleteCDSCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.28 CFE_ES_DeleteCDSCmd_Payload_t Struct Reference

Delete Critical Data Store Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- char CdsName [CFE_MISSION_ES_CDS_MAX_NAME_LEN]

  ASCII text string containing name of CDS to delete.

12.28.1 Detailed Description

For command details, see CFE_ES_DELETE_CDS_CC

Definition at line 1321 of file cfe_es_msg.h.

12.28.2 Field Documentation

12.28.2.1 CdsName

```
char CFE_ES_DeleteCDSCmd_Payload_t::CdsName[CFE_MISSION_ES_CDS_MAX_NAME_LEN]
```

Definition at line 1323 of file cfe_es_msg.h.

Referenced by CFE_ES_DeleteCDSCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.29 CFE_ES_DumpCDSRegistry_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_DumpCDSRegistryCmd_Payload_t Payload

12.29.1 Detailed Description

Definition at line 1438 of file cfe_es_msg.h.

12.29.2 Field Documentation

12.29.2.1 CmdHeader

```
uint8 CFE_ES_DumpCDSRegistry_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1440 of file cfe_es_msg.h.

12.29.2.2 Payload

```
CFE_ES_DumpCDSRegistryCmd_Payload_t CFE_ES_DumpCDSRegistry_t::Payload
```

Definition at line 1441 of file cfe_es_msg.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.30 CFE_ES_DumpCDSRegistryCmd_Payload_t Struct Reference

Dump CDS Registry Command.

`#include <cfe_es_msg.h>`

Data Fields

- char DumpFilename [CFE_MISSION_MAX_PATH_LEN]

  ASCII text string of full path and filename of file CDS Registry is to be written.

12.30.1 Detailed Description

For command details, see CFE_ES_DUMP_CDS_REGISTRY_CC

Definition at line 1432 of file cfe_es_msg.h.

12.30.2 Field Documentation

12.30.2.1 DumpFilename

`char CFE_ES_DumpCDSRegistryCmd_Payload_t::DumpFilename[CFE_MISSION_MAX_PATH_LEN]`

Definition at line 1434 of file cfe_es_msg.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.31 CFE_ES_ERLog_t Struct Reference

`#include <cfe_es_erlog_typedef.h>`

Data Fields

- uint32 LogEntryType
- uint32 ResetType
- uint32 ResetSubtype
- uint32 BootSource
- uint32 ProcessorResetCount
- uint32 MaxProcessorResetCount
- CFE_ES_DebugVariables_t DebugVars
- CFE_TIME_SysTime_t TimeCode
- char Description [80]
- uint32 ContextSize
- uint32 AppID
- uint32 Context [CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE/sizeof(uint32)]

12.31.1 Detailed Description

Definition at line 52 of file cfe_es_erlog_typedef.h.

12.31.2 Field Documentation

12.31.2.1 AppID

```
uint32 CFE_ES_ERLog_t::AppID
```

Definition at line 64 of file cfe_es_erlog_typedef.h.

12.31.2.2 BootSource

```
uint32 CFE_ES_ERLog_t::BootSource
```

Definition at line 57 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.3 Context

```
uint32 CFE_ES_ERLog_t::Context[CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE/sizeof(uint32)]
```

Definition at line 65 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.4 ContextSize

```
uint32 CFE_ES_ERLog_t::ContextSize
```

Definition at line 63 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.5 DebugVars

```
CFE_ES_DebugVariables_t CFE_ES_ERLog_t::DebugVars
```

Definition at line 60 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.6 Description

```
char CFE_ES_ERLog_t::Description[80]
```

Definition at line 62 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.7 LogEntryType

```
uint32 CFE_ES_ERLog_t::LogEntryType
```

Definition at line 54 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.8 MaxProcessorResetCount

```
uint32 CFE_ES_ERLog_t::MaxProcessorResetCount
```

Definition at line 59 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.9 ProcessorResetCount

```
uint32 CFE_ES_ERLog_t::ProcessorResetCount
```

Definition at line 58 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.10 ResetSubtype

```
uint32 CFE_ES_ERLog_t::ResetSubtype
```

Definition at line 56 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.11 ResetType

```
uint32 CFE_ES_ERLog_t::ResetType
```

Definition at line 55 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

12.31.2.12 TimeCode

```
CFE_TIME_SysTime_t CFE_ES_ERLog_t::TimeCode
```

Definition at line 61 of file cfe_es_erlog_typedef.h.

Referenced by CFE_ES_WriteToERLog().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_erlog_typedef.h

12.32 CFE_ES_FileNameCmd_Payload_t Struct Reference

Payload format for commands which accept a single file name.

```
#include <cfe_es_msg.h>
```

Data Fields

- char FileName [CFE_MISSION_MAX_PATH_LEN]

  ASCII text string containing full path and filename of file in which Application data is to be dumped.

12.32.1 Detailed Description

This format is shared by several executive services commands. For command details, see CFE_ES_QUERY_ALL_CC, CFE_ES_QUERY_ALL_TASKS_CC, CFE_ES_WRITE_SYSLOG_CC, and CFE_ES_WRITE_ER_LOG_CC

Definition at line 1183 of file cfe_es_msg.h.

12.32.2 Field Documentation

12.32.2.1 FileName

```
char CFE_ES_FileNameCmd_Payload_t::FileName[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 1185 of file cfe_es_msg.h.

Referenced by CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_WriteERLogCmd(), and CFE_ES↩ _WriteSyslogCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.33 CFE_ES_FileNameCmd_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

  cFE Software Bus Command Message Header
- CFE_ES_FileNameCmd_Payload_t Payload

12.33.1 Detailed Description

Definition at line 1189 of file cfe_es_msg.h.

12.33.2 Field Documentation

12.33.2.1 CmdHeader

```
uint8 CFE_ES_FileNameCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1191 of file cfe_es_msg.h.

12.33.2.2 Payload

```
CFE_ES_FileNameCmd_Payload_t CFE_ES_FileNameCmd_t::Payload
```

Definition at line 1192 of file cfe_es_msg.h.

Referenced by CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_WriteERLogCmd(), and CFE_ES↩
_WriteSyslogCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.34 CFE_ES_FuncPtrUnion_t Union Reference

```
#include <cfe_es_start.h>
```

Data Fields

- CFE_ES_EarlyInitFuncPtr_t FunctionPtr
- CFE_ES_MainAppFuncPtr_t MainAppPtr
- void * VoidPtr

12.34.1 Detailed Description

Definition at line 67 of file cfe_es_start.h.

12.34.2 Field Documentation

12.34.2.1 FunctionPtr

```
CFE_ES_EarlyInitFuncPtr_t CFE_ES_FuncPtrUnion_t::FunctionPtr
```

Definition at line 69 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

12.34.2.2 MainAppPtr

```
CFE_ES_MainAppFuncPtr_t CFE_ES_FuncPtrUnion_t::MainAppPtr
```

Definition at line 70 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

12.34.2.3 VoidPtr

```
void* CFE_ES_FuncPtrUnion_t::VoidPtr
```

Definition at line 71 of file cfe_es_start.h.

The documentation for this union was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_start.h

12.35 CFE_ES_GenCounterRecord_t Struct Reference

```
#include <cfe_es_global.h>
```

Data Fields

- bool RecordUsed
- uint32 Counter
- char CounterName [OS_MAX_API_NAME]

12.35.1 Detailed Description

Definition at line 65 of file cfe_es_global.h.

12.35.2 Field Documentation

12.35.2.1 Counter

```
uint32 CFE_ES_GenCounterRecord_t::Counter
```

Definition at line 68 of file cfe_es_global.h.

Referenced by CFE_ES_DeleteGenCounter(), CFE_ES_GetGenCount(), CFE_ES_IncrementGenCounter(), CFE_ES_RegisterGenCounter(), and CFE_ES_SetGenCount().

12.35.2.2 CounterName

```
char CFE_ES_GenCounterRecord_t::CounterName[OS_MAX_API_NAME]
```

Definition at line 69 of file cfe_es_global.h.

Referenced by CFE_ES_GetGenCounterIDByName(), and CFE_ES_RegisterGenCounter().

12.35.2.3 RecordUsed

```
bool CFE_ES_GenCounterRecord_t::RecordUsed
```

Definition at line 67 of file cfe_es_global.h.

Referenced by CFE_ES_DeleteGenCounter(), CFE_ES_GetGenCount(), CFE_ES_GetGenCounterIDByName(), CFE_ES_IncrementGenCounter(), CFE_ES_Main(), CFE_ES_RegisterGenCounter(), and CFE_ES_SetGenCount().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_global.h

12.36 CFE_ES_Global_t Struct Reference

```
#include <cfe_es_global.h>
```

Data Fields

- CFE_ES_DebugVariables_t DebugVars
- uint32 SharedDataMutex
- uint32 SystemState
- uint32 RegisteredTasks
- CFE_ES_TaskRecord_t TaskTable [OS_MAX_TASKS]
- uint32 RegisteredCoreApps
- uint32 RegisteredExternalApps
- CFE_ES_AppRecord_t AppTable [CFE_PLATFORM_ES_MAX_APPLICATIONS]
- uint32 RegisteredLibs
- CFE_ES_LibRecord_t LibTable [CFE_PLATFORM_ES_MAX_LIBRARIES]
- CFE_ES_GenCounterRecord_t CounterTable [CFE_PLATFORM_ES_MAX_GEN_COUNTERS]
- CFE_ES_CDSVariables_t CDSVars

12.36.1 Detailed Description

Definition at line 78 of file cfe_es_global.h.

12.36.2 Field Documentation

12.36.2.1 AppTable

```
CFE_ES_AppRecord_t CFE_ES_Global_t::AppTable[CFE_PLATFORM_ES_MAX_APPLICATIONS]
```

Definition at line 106 of file cfe_es_global.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateChildTask(), CFE_ES_Create↩Objects(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CF↩E_ES_GetAppIDByName(), CFE_ES_GetAppInfo(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), C↩FE_ES_GetTaskInfo(), CFE_ES_ListApplications(), CFE_ES_Main(), CFE_ES_MainTaskSyncDelay(), CFE_ES↩_ProcessControlRequest(), CFE_ES_ProcessCoreException(), CFE_ES_QueryAllCmd(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_ScanAppTable(), CFE_ES_SetAppState(), and CFE_ES↩_WaitForSystemState().

12.36.2.2 CDSVars

```
CFE_ES_CDSVariables_t CFE_ES_Global_t::CDSVars
```

Definition at line 122 of file cfe_es_global.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CopyTo↩CDS(), CFE_ES_DeleteCDS(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_FindCDSInRegistry(), CFE_ES_Find↩FreeCDSRegistryEntry(), CFE_ES_InitCDSRegistry(), CFE_ES_InitializeCDS(), CFE_ES_LockCDSRegistry(), CF↩E_ES_PutCDSBlock(), CFE_ES_RebuildCDS(), CFE_ES_RegisterCDS(), CFE_ES_RegisterCDSEx(), CFE_ES_↩RestoreFromCDS(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UpdateCDSRegistry(), and CFE_ES_ValidateCDS().

12.36.2.3 CounterTable

```
CFE_ES_GenCounterRecord_t CFE_ES_Global_t::CounterTable[CFE_PLATFORM_ES_MAX_GEN_COUNTERS]
```

Definition at line 117 of file cfe_es_global.h.

Referenced by CFE_ES_DeleteGenCounter(), CFE_ES_GetGenCount(), CFE_ES_GetGenCounterIDByName(), CFE_ES_IncrementGenCounter(), CFE_ES_Main(), CFE_ES_RegisterGenCounter(), and CFE_ES_SetGenCount().

12.36.2.4 DebugVars

```
CFE_ES_DebugVariables_t CFE_ES_Global_t::DebugVars
```

Definition at line 83 of file cfe_es_global.h.

Referenced by CFE_ES_SetupResetVariables(), and CFE_ES_WriteToERLog().

12.36.2.5 LibTable

```
CFE_ES_LibRecord_t CFE_ES_Global_t::LibTable[CFE_PLATFORM_ES_MAX_LIBRARIES]
```

Definition at line 112 of file cfe_es_global.h.

Referenced by CFE_ES_LoadLibrary().

12.36.2.6 RegisteredCoreApps

```
uint32 CFE_ES_Global_t::RegisteredCoreApps
```

Definition at line 104 of file cfe_es_global.h.

Referenced by CFE_ES_CreateObjects(), and CFE_ES_HousekeepingCmd().

12.36.2.7 RegisteredExternalApps

```
uint32 CFE_ES_Global_t::RegisteredExternalApps
```

Definition at line 105 of file cfe_es_global.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), and CFE_ES_HousekeepingCmd().

12.36.2.8 RegisteredLibs

```
uint32 CFE_ES_Global_t::RegisteredLibs
```

Definition at line 111 of file cfe_es_global.h.

Referenced by CFE_ES_HousekeepingCmd(), and CFE_ES_LoadLibrary().

12.36.2.9 RegisteredTasks

```
uint32 CFE_ES_Global_t::RegisteredTasks
```

Definition at line 98 of file cfe_es_global.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanupTaskResources(), CFE_ES_CreateChildTask(), CFE_ES_↩ CreateObjects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), and CFE_ES_HousekeepingCmd().

12.36.2.10 SharedDataMutex

```
uint32 CFE_ES_Global_t::SharedDataMutex
```

Definition at line 88 of file cfe_es_global.h.

Referenced by CFE_ES_LockSharedData(), CFE_ES_Main(), and CFE_ES_UnlockSharedData().

12.36.2.11 SystemState

```
uint32 CFE_ES_Global_t::SystemState
```

Definition at line 93 of file cfe_es_global.h.

Referenced by CFE_ES_Main(), and CFE_ES_WaitForSystemState().

12.36.2.12 TaskTable

```
CFE_ES_TaskRecord_t CFE_ES_Global_t::TaskTable[OS_MAX_TASKS]
```

Definition at line 99 of file cfe_es_global.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CleanupTaskResources(), CFE_ES_↩ CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), CFE_ES_Get↩ AppIDInternal(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetTaskInfo(), CFE_ES_IncrementTaskCounter(), CFE_E↩ S_ListTasks(), CFE_ES_Main(), CFE_ES_ProcessCoreException(), CFE_ES_QueryAllTasksCmd(), and CFE_ES_↩ RunLoop().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_global.h

12.37 CFE_ES_HousekeepingTlm_Payload_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CommandCounter

  The ES Application Command Counter.
- uint8 CommandErrorCounter

  The ES Application Command Error Counter.
- uint16 CFECoreChecksum

  Checksum of cFE Core Code.
- uint8 CFEMajorVersion

  Major Version Number of cFE.
- uint8 CFEMinorVersion

  Minor Version Number of cFE.
- uint8 CFERevision

  Sub-Minor Version Number of cFE.
- uint8 CFEMissionRevision

  Mission Version Number of cFE.
- uint8 OSALMajorVersion

  OS Abstraction Layer Major Version Number.
- uint8 OSALMinorVersion

  OS Abstraction Layer Minor Version Number.
- uint8 OSALRevision

  OS Abstraction Layer Revision Number.
- uint8 OSALMissionRevision

  OS Abstraction Layer MissionRevision Number.
- uint32 SysLogBytesUsed

  Total number of bytes used in system log.
- uint32 SysLogSize

  Total size of the system log.
- uint32 SysLogEntries

  Number of entries in the system log.
- uint32 SysLogMode

  Write/Overwrite Mode.
- uint32 ERLogIndex

  Current index of the ER Log (wraps around)
- uint32 ERLogEntries

  Number of entries made in the ER Log since the power on.
- uint32 RegisteredCoreApps

  Number of Applications registered with ES.
- uint32 RegisteredExternalApps

  Number of Applications registered with ES.
- uint32 RegisteredTasks

  Number of Tasks (main AND child tasks) registered with ES.

- uint32 RegisteredLibs

  Number of Libraries registered with ES.
- uint32 ResetType

  Reset type (PROCESSOR or POWERON)
- uint32 ResetSubtype

  Reset Sub Type.
- uint32 ProcessorResets

  Number of processor resets since last power on.
- uint32 MaxProcessorResets

  Max processor resets before a power on is done.
- uint32 BootSource

  Boot source (as provided from BSP)
- uint32 PerfState

  Current state of Performance Analyzer.
- uint32 PerfMode

  Current mode of Performance Analyzer.
- uint32 PerfTriggerCount

  Number of Times Perfomance Analyzer has Triggered.
- uint32 PerfFilterMask [CFE_MISSION_ES_PERF_MAX_IDS/32]

  Current Setting of Performance Analyzer Filter Masks.
- uint32 PerfTriggerMask [CFE_MISSION_ES_PERF_MAX_IDS/32]

  Current Setting of Performance Analyzer Trigger Masks.
- uint32 PerfDataStart

  Identifies First Stored Entry in Performance Analyzer Log.
- uint32 PerfDataEnd

  Identifies Last Stored Entry in Performance Analyzer Log.
- uint32 PerfDataCount

  Number of Entries Put Into the Performance Analyzer Log.
- uint32 PerfDataToWrite

  Number of Performance Analyzer Log Entries Left to be Written to Log Dump File.
- uint32 HeapBytesFree

  Number of free bytes remaining in the OS heap.
- uint32 HeapBlocksFree

  Number of free blocks remaining in the OS heap.
- uint32 HeapMaxBlockSize

  Number of bytes in the largest free block.

12.37.1 Detailed Description

Name Executive Services Housekeeping Packet

Definition at line 1485 of file cfe_es_msg.h.

12.37.2 Field Documentation

12.37.2.1 BootSource

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::BootSource
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_BootSource**

Definition at line 1542 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.2 CFECoreChecksum

```
uint16 CFE_ES_HousekeepingTlm_Payload_t::CFECoreChecksum
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CKSUM**

Definition at line 1492 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.3 CFEMajorVersion

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::CFEMajorVersion
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CFEMAJORVER**

Definition at line 1494 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.4 CFEMinorVersion

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::CFEMinorVersion
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CFEMINORVER**

Definition at line 1496 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.5 CFEMissionRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::CFEMissionRevision
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CFEMISSIONREV**

Definition at line 1500 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.6 CFERevision

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::CFERevision
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CFEREVISION**

Definition at line 1498 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.7 CommandCounter

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::CommandCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CMDPC**

Definition at line 1487 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.8 CommandErrorCounter

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::CommandErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_CMDEC**

Definition at line 1489 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.9 ERLogEntries

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::ERLogEntries
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ERLOGENTRIES**

Definition at line 1522 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.10 ERLogIndex

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::ERLogIndex
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ERLOGINDEX**

Definition at line 1520 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.11 HeapBlocksFree

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::HeapBlocksFree
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_HeapBlocksFree**

Definition at line 1565 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.12 HeapBytesFree

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::HeapBytesFree
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_HeapBytesFree**

Definition at line 1563 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.13 HeapMaxBlockSize

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::HeapMaxBlockSize
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_HeapMaxBlkSize**

Definition at line 1567 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.14 MaxProcessorResets

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::MaxProcessorResets
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_MaxProcResets**

Definition at line 1540 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.15 OSALMajorVersion

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::OSALMajorVersion
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_OSMAJORVER**

Definition at line 1502 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.16 OSALMinorVersion

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::OSALMinorVersion
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_OSMINORVER**

Definition at line 1504 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.17 OSALMissionRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::OSALMissionRevision
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_OSMISSIONREV**

Definition at line 1508 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.18 OSALRevision

```
uint8 CFE_ES_HousekeepingTlm_Payload_t::OSALRevision
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_OSREVISION**

Definition at line 1506 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskInit().

12.37.2.19 PerfDataCount

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfDataCount
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfDataCnt**

Definition at line 1559 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.20 PerfDataEnd

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfDataEnd
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfDataEnd**

Definition at line 1557 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.21 PerfDataStart

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfDataStart
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfDataStart**

Definition at line 1555 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.22 PerfDataToWrite

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfDataToWrite
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfData2Write**

Definition at line 1561 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.23 PerfFilterMask

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfFilterMask[CFE_MISSION_ES_PERF_MAX_IDS/32]
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfFltrMask[MaskCnt]**

Definition at line 1551 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.24 PerfMode

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfMode
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfMode**

Definition at line 1547 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.25 PerfState

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfState
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfState**

Definition at line 1545 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.26 PerfTriggerCount

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfTriggerCount
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfTrigCnt**

Definition at line 1549 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.27 PerfTriggerMask

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::PerfTriggerMask[CFE_MISSION_ES_PERF_MAX_IDS/32]
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PerfTrigMask[MaskCnt]**

Definition at line 1553 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.28 ProcessorResets

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::ProcessorResets
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_ProcResetCnt**

Definition at line 1538 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.29 RegisteredCoreApps

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::RegisteredCoreApps
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_RegCoreApps**

Definition at line 1525 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.30 RegisteredExternalApps

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::RegisteredExternalApps
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_RegExtApps**

Definition at line 1527 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.31 RegisteredLibs

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::RegisteredLibs
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_RegLibs**

Definition at line 1531 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.32 RegisteredTasks

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::RegisteredTasks
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_RegTasks**

Definition at line 1529 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.33 ResetSubtype

uint32 CFE_ES_HousekeepingTlm_Payload_t::ResetSubtype

Telemetry Mnemonic(s) **$sc_$cpu_ES_ResetSubtype**

Definition at line 1536 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.34 ResetType

uint32 CFE_ES_HousekeepingTlm_Payload_t::ResetType

Telemetry Mnemonic(s) **$sc_$cpu_ES_ResetType**

Definition at line 1534 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.35 SysLogBytesUsed

uint32 CFE_ES_HousekeepingTlm_Payload_t::SysLogBytesUsed

Telemetry Mnemonic(s) **$sc_$cpu_ES_SYSLOGBYTEUSED**

Definition at line 1511 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.36 SysLogEntries

uint32 CFE_ES_HousekeepingTlm_Payload_t::SysLogEntries

Telemetry Mnemonic(s) **$sc_$cpu_ES_SYSLOGENTRIES**

Definition at line 1515 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd(), and CFE_ES_SysLogDump().

12.37.2.37 SysLogMode

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::SysLogMode
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_SYSLOGMODE**

Definition at line 1517 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

12.37.2.38 SysLogSize

```
uint32 CFE_ES_HousekeepingTlm_Payload_t::SysLogSize
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_SYSLOGSIZE**

Definition at line 1513 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.38 CFE_ES_HousekeepingTlm_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
  cFE Software Bus Telemetry Message Header
- CFE_ES_HousekeepingTlm_Payload_t Payload

12.38.1 Detailed Description

Definition at line 1571 of file cfe_es_msg.h.

12.38.2 Field Documentation

12.38.2.1 Payload

```
CFE_ES_HousekeepingTlm_Payload_t CFE_ES_HousekeepingTlm_t::Payload
```

Definition at line 1574 of file cfe_es_msg.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_SysLogDump(), and CFE_ES_TaskInit().

12.38.2.2 TlmHeader

```
uint8 CFE_ES_HousekeepingTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1573 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.39 CFE_ES_LibRecord_t Struct Reference

```
#include <cfe_es_apps.h>
```

Data Fields

- bool RecordUsed
- char LibName [OS_MAX_API_NAME]

12.39.1 Detailed Description

Definition at line 132 of file cfe_es_apps.h.

12.39.2 Field Documentation

12.39.2.1 LibName

```
char CFE_ES_LibRecord_t::LibName[OS_MAX_API_NAME]
```

Definition at line 135 of file cfe_es_apps.h.

Referenced by CFE_ES_LoadLibrary().

12.39.2.2 RecordUsed

```
bool CFE_ES_LibRecord_t::RecordUsed
```

Definition at line 134 of file cfe_es_apps.h.

Referenced by CFE_ES_LoadLibrary().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.h

12.40 CFE_ES_MainTaskInfo_t Struct Reference

```
#include <cfe_es_apps.h>
```

Data Fields

- uint32 MainTaskId
- char MainTaskName [OS_MAX_API_NAME]

12.40.1 Detailed Description

Definition at line 91 of file cfe_es_apps.h.

12.40.2 Field Documentation

12.40.2.1 MainTaskId

```
uint32 CFE_ES_MainTaskInfo_t::MainTaskId
```

Definition at line 93 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateChildTask(), CFE_ES_Create↩Objects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), CFE_ES_GetAppInfoInternal(), and CFE_ES_Run↩Loop().

12.40.2.2 MainTaskName

```
char CFE_ES_MainTaskInfo_t::MainTaskName[OS_MAX_API_NAME]
```

Definition at line 94 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateObjects(), and CFE_ES_GetAppInfoInternal().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.h

12.41 CFE_ES_MemPoolStats_t Struct Reference

```
#include <cfe_es.h>
```

Data Fields

- uint32 PoolSize
  Size of Memory Pool (in bytes)
- uint32 NumBlocksRequested
  Number of times a memory block has been allocated.
- uint32 CheckErrCtr
  Number of errors detected when freeing a memory block.
- uint32 NumFreeBytes
  Number of bytes never allocated to a block.
- CFE_ES_BlockStats_t BlockStats [CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]
  Contains stats on each block size.

12.41.1 Detailed Description

Definition at line 278 of file cfe_es.h.

12.41.2 Field Documentation

12.41.2.1 BlockStats

```
CFE_ES_BlockStats_t CFE_ES_MemPoolStats_t::BlockStats[CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]
```

Telemetry Mnemonic(s) **`$sc_$cpu_ES_BlkStats[BLK_SIZES]`**

Definition at line 288 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

12.41.2.2 CheckErrCtr

```
uint32 CFE_ES_MemPoolStats_t::CheckErrCtr
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_BlkErrCTR**

Definition at line 284 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

12.41.2.3 NumBlocksRequested

```
uint32 CFE_ES_MemPoolStats_t::NumBlocksRequested
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_BlksREQ**

Definition at line 282 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

12.41.2.4 NumFreeBytes

```
uint32 CFE_ES_MemPoolStats_t::NumFreeBytes
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_FreeBytes**

Definition at line 286 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

12.41.2.5 PoolSize

```
uint32 CFE_ES_MemPoolStats_t::PoolSize
```

Telemetry Mnemonic(s) **$sc_$cpu_ES_PoolSize**

Definition at line 280 of file cfe_es.h.

Referenced by CFE_ES_GetMemPoolStats().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es.h

12.42 CFE_ES_MemStatsTlm_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
  cFE Software Bus Telemetry Message Header
- CFE_ES_PoolStatsTlm_Payload_t Payload

12.42.1 Detailed Description

Definition at line 1474 of file cfe_es_msg.h.

12.42.2 Field Documentation

12.42.2.1 Payload

`CFE_ES_PoolStatsTlm_Payload_t CFE_ES_MemStatsTlm_t::Payload`

Definition at line 1477 of file cfe_es_msg.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

12.42.2.2 TlmHeader

`uint8 CFE_ES_MemStatsTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]`

Definition at line 1476 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.43 CFE_ES_NoArgsCmd_t Struct Reference

Generic "no arguments" command.

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header

12.43.1 Detailed Description

This command structure is used for commands that do not have any parameters. This includes:

1. The Housekeeping Request Message
2. The No-Op Command (For details, see CFE_ES_NOOP_CC)
3. The Reset Counters Command (For details, see CFE_ES_RESET_COUNTERS_CC)

Definition at line 1118 of file cfe_es_msg.h.

12.43.2 Field Documentation

12.43.2.1 CmdHeader

```
uint8 CFE_ES_NoArgsCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1120 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.44 CFE_ES_ObjectTable_t Struct Reference

```
#include <cfe_es_start.h>
```

Data Fields

- uint32 ObjectType
- char ObjectName [OS_MAX_API_NAME]
- CFE_ES_FuncPtrUnion_t FuncPtrUnion
- uint32 ObjectPriority
- uint32 ObjectSize
- uint32 ObjectFlags

12.44.1 Detailed Description

Definition at line 74 of file cfe_es_start.h.

12.44.2 Field Documentation

12.44.2.1 FuncPtrUnion

```
CFE_ES_FuncPtrUnion_t CFE_ES_ObjectTable_t::FuncPtrUnion
```

Definition at line 78 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

12.44.2.2 ObjectFlags

```
uint32 CFE_ES_ObjectTable_t::ObjectFlags
```

Definition at line 81 of file cfe_es_start.h.

12.44.2.3 ObjectName

```
char CFE_ES_ObjectTable_t::ObjectName[OS_MAX_API_NAME]
```

Definition at line 77 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

12.44.2.4 ObjectPriority

```
uint32 CFE_ES_ObjectTable_t::ObjectPriority
```

Definition at line 79 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

12.44.2.5 ObjectSize

```
uint32 CFE_ES_ObjectTable_t::ObjectSize
```

Definition at line 80 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

12.44.2.6 ObjectType

```
uint32 CFE_ES_ObjectTable_t::ObjectType
```

Definition at line 76 of file cfe_es_start.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_start.h

12.45 CFE_ES_OneAppTlm_Payload_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- CFE_ES_AppInfo_t AppInfo

  For more information, see CFE_ES_AppInfo_t.

12.45.1 Detailed Description

Name Single Application Information Packet

Definition at line 1452 of file cfe_es_msg.h.

12.45.2 Field Documentation

12.45.2.1 AppInfo

```
CFE_ES_AppInfo_t CFE_ES_OneAppTlm_Payload_t::AppInfo
```

Definition at line 1454 of file cfe_es_msg.h.

Referenced by CFE_ES_QueryOneCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.46 CFE_ES_OneAppTlm_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]

  cFE Software Bus Telemetry Message Header
- CFE_ES_OneAppTlm_Payload_t Payload

12.46.1 Detailed Description

Definition at line 1458 of file cfe_es_msg.h.

12.46.2 Field Documentation

12.46.2.1 Payload

```
CFE_ES_OneAppTlm_Payload_t CFE_ES_OneAppTlm_t::Payload
```

Definition at line 1461 of file cfe_es_msg.h.

Referenced by CFE_ES_QueryOneCmd().

12.46.2.2 TlmHeader

```
uint8 CFE_ES_OneAppTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1460 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.47 CFE_ES_OverWriteSyslog_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_OverWriteSysLogCmd_Payload_t Payload

12.47.1 Detailed Description

Definition at line 1217 of file cfe_es_msg.h.

12.47.2 Field Documentation

12.47.2.1 CmdHeader

```
uint8 CFE_ES_OverWriteSyslog_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1219 of file cfe_es_msg.h.

12.47.2.2 Payload

```
CFE_ES_OverWriteSysLogCmd_Payload_t CFE_ES_OverWriteSyslog_t::Payload
```

Definition at line 1220 of file cfe_es_msg.h.

Referenced by CFE_ES_OverWriteSyslogCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.48 CFE_ES_OverWriteSysLogCmd_Payload_t Struct Reference

Overwrite/Discard System Log Configuration Command.

`#include <cfe_es_msg.h>`

Data Fields

- uint32 Mode

  CFE_ES_LogMode_DISCARD=Throw away most recent messages, CFE_ES_LogMode_OVERWRITE=Overwrite oldest with most recent

12.48.1 Detailed Description

For command details, see CFE_ES_OVER_WRITE_SYSLOG_CC

Definition at line 1210 of file cfe_es_msg.h.

12.48.2 Field Documentation

12.48.2.1 Mode

`uint32 CFE_ES_OverWriteSysLogCmd_Payload_t::Mode`

Definition at line 1212 of file cfe_es_msg.h.

Referenced by CFE_ES_OverWriteSyslogCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.49 CFE_ES_PerfData_t Struct Reference

`#include <cfe_es_perfdata_typedef.h>`

Data Fields

- CFE_ES_PerfMetaData_t MetaData
- CFE_ES_PerfDataEntry_t DataBuffer [CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE]

12.49.1 Detailed Description

Definition at line 64 of file cfe_es_perfdata_typedef.h.

12.49.2 Field Documentation

12.49.2.1 DataBuffer

```
CFE_ES_PerfDataEntry_t CFE_ES_PerfData_t::DataBuffer[CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE]
```

Definition at line 66 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_PerfLogAdd(), and CFE_ES_PerfLogDump().

12.49.2.2 MetaData

```
CFE_ES_PerfMetaData_t CFE_ES_PerfData_t::MetaData
```

Definition at line 65 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_PerfLogDump(), CFE_ES_SetPerf↩FilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_SetupPerfVariables(), CFE_ES_StartPerfDataCmd(), and CFE_ES_StopPerfDataCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_perfdata_typedef.h

12.50 CFE_ES_PerfDataEntry_t Struct Reference

```
#include <cfe_es_perfdata_typedef.h>
```

Data Fields

- uint32 Data
- uint32 TimerUpper32
- uint32 TimerLower32

12.50.1 Detailed Description

Definition at line 40 of file cfe_es_perfdata_typedef.h.

12.50.2 Field Documentation

12.50.2.1 Data

```
uint32 CFE_ES_PerfDataEntry_t::Data
```

Definition at line 41 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_PerfLogAdd().

12.50.2.2 TimerLower32

```
uint32 CFE_ES_PerfDataEntry_t::TimerLower32
```

Definition at line 43 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_PerfLogAdd().

12.50.2.3 TimerUpper32

```
uint32 CFE_ES_PerfDataEntry_t::TimerUpper32
```

Definition at line 42 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_PerfLogAdd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_perfdata_typedef.h

12.51 CFE_ES_PerfLogDump_t Struct Reference

```
#include <cfe_es_perf.h>
```

Data Fields

- uint32 DataToWrite
- uint32 ChildID
- char DataFileName [OS_MAX_PATH_LEN]
- int32 DataFileDescriptor

12.51.1 Detailed Description

Definition at line 71 of file cfe_es_perf.h.

12.51.2 Field Documentation

12.51.2.1 ChildID

```
uint32 CFE_ES_PerfLogDump_t::ChildID
```

Definition at line 73 of file cfe_es_perf.h.

Referenced by CFE_ES_SetupPerfVariables(), and CFE_ES_StopPerfDataCmd().

12.51.2.2 DataFileDescriptor

```
int32 CFE_ES_PerfLogDump_t::DataFileDescriptor
```

Definition at line 75 of file cfe_es_perf.h.

Referenced by CFE_ES_PerfLogDump(), and CFE_ES_StopPerfDataCmd().

12.51.2.3 DataFileName

```
char CFE_ES_PerfLogDump_t::DataFileName[OS_MAX_PATH_LEN]
```

Definition at line 74 of file cfe_es_perf.h.

Referenced by CFE_ES_PerfLogDump(), CFE_ES_SetupPerfVariables(), and CFE_ES_StopPerfDataCmd().

12.51.2.4 DataToWrite

```
uint32 CFE_ES_PerfLogDump_t::DataToWrite
```

Definition at line 72 of file cfe_es_perf.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogDump(), CFE_ES_SetupPerfVariables(), CFE_ES_↩ StartPerfDataCmd(), and CFE_ES_StopPerfDataCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_perf.h

12.52 CFE_ES_PerfMetaData_t Struct Reference

`#include <cfe_es_perfdata_typedef.h>`

Data Fields

- uint8 Version
- uint8 Endian
- uint8 Spare [2]
- uint32 TimerTicksPerSecond
- uint32 TimerLow32Rollover
- uint32 State
- uint32 Mode
- uint32 TriggerCount
- uint32 DataStart
- uint32 DataEnd
- uint32 DataCount
- uint32 InvalidMarkerReported
- uint32 FilterTriggerMaskSize
- uint32 FilterMask [CFE_ES_PERF_32BIT_WORDS_IN_MASK]
- uint32 TriggerMask [CFE_ES_PERF_32BIT_WORDS_IN_MASK]

12.52.1 Detailed Description

Definition at line 46 of file cfe_es_perfdata_typedef.h.

12.52.2 Field Documentation

12.52.2.1 DataCount

`uint32 CFE_ES_PerfMetaData_t::DataCount`

Definition at line 57 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_PerfLogDump(), CFE_ES_Setup↩PerfVariables(), CFE_ES_StartPerfDataCmd(), and CFE_ES_StopPerfDataCmd().

12.52.2.2 DataEnd

`uint32 CFE_ES_PerfMetaData_t::DataEnd`

Definition at line 56 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetupPerfVariables(), and CFE_E↩S_StartPerfDataCmd().

12.52.2.3 DataStart

```
uint32 CFE_ES_PerfMetaData_t::DataStart
```

Definition at line 55 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetupPerfVariables(), and CFE_ES_StartPerfDataCmd().

12.52.2.4 Endian

```
uint8 CFE_ES_PerfMetaData_t::Endian
```

Definition at line 48 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_SetupPerfVariables().

12.52.2.5 FilterMask

```
uint32 CFE_ES_PerfMetaData_t::FilterMask[CFE_ES_PERF_32BIT_WORDS_IN_MASK]
```

Definition at line 60 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetPerfFilterMaskCmd(), and CFE_ES_SetupPerfVariables().

12.52.2.6 FilterTriggerMaskSize

```
uint32 CFE_ES_PerfMetaData_t::FilterTriggerMaskSize
```

Definition at line 59 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_SetupPerfVariables().

12.52.2.7 InvalidMarkerReported

```
uint32 CFE_ES_PerfMetaData_t::InvalidMarkerReported
```

Definition at line 58 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_PerfLogAdd(), CFE_ES_SetupPerfVariables(), and CFE_ES_StartPerfDataCmd().

12.52.2.8 Mode

uint32 CFE_ES_PerfMetaData_t::Mode

Definition at line 53 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetupPerfVariables(), and CFE_ES_StartPerfDataCmd().

12.52.2.9 Spare

uint8 CFE_ES_PerfMetaData_t::Spare[2]

Definition at line 49 of file cfe_es_perfdata_typedef.h.

12.52.2.10 State

uint32 CFE_ES_PerfMetaData_t::State

Definition at line 52 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetupPerfVariables(), CFE_ES_StartPerfDataCmd(), and CFE_ES_StopPerfDataCmd().

12.52.2.11 TimerLow32Rollover

uint32 CFE_ES_PerfMetaData_t::TimerLow32Rollover

Definition at line 51 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_SetupPerfVariables().

12.52.2.12 TimerTicksPerSecond

uint32 CFE_ES_PerfMetaData_t::TimerTicksPerSecond

Definition at line 50 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_SetupPerfVariables().

12.52.2.13 TriggerCount

```
uint32 CFE_ES_PerfMetaData_t::TriggerCount
```

Definition at line 54 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetupPerfVariables(), and CFE_ES_StartPerfDataCmd().

12.52.2.14 TriggerMask

```
uint32 CFE_ES_PerfMetaData_t::TriggerMask[CFE_ES_PERF_32BIT_WORDS_IN_MASK]
```

Definition at line 61 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_PerfLogAdd(), CFE_ES_SetPerfTriggerMaskCmd(), and CFE_ES_SetupPerfVariables().

12.52.2.15 Version

```
uint8 CFE_ES_PerfMetaData_t::Version
```

Definition at line 47 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_SetupPerfVariables().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_perfdata_typedef.h

12.53 CFE_ES_PoolAlign_t Union Reference

```
#include <cfe_es.h>
```

Data Fields

- void * Ptr
- long long int LongInt
- long double LongDouble

12.53.1 Detailed Description

Union that can be used for minimum memory alignment of ES memory pools on the target. It contains the longest native data types such that the alignment of this structure should reflect the largest possible alignment requirements for any data on this processor.

Definition at line 317 of file cfe_es.h.

12.53.2 Field Documentation

12.53.2.1 LongDouble

```
long double CFE_ES_PoolAlign_t::LongDouble
```

Definition at line 322 of file cfe_es.h.

12.53.2.2 LongInt

```
long long int CFE_ES_PoolAlign_t::LongInt
```

Definition at line 321 of file cfe_es.h.

12.53.2.3 Ptr

```
void* CFE_ES_PoolAlign_t::Ptr
```

Definition at line 319 of file cfe_es.h.

The documentation for this union was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es.h

12.54 CFE_ES_PoolStatsTlm_Payload_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- CFE_ES_MemHandle_t PoolHandle

  Handle of memory pool whose stats are being telemetered.
- CFE_ES_MemPoolStats_t PoolStats

  For more info, see CFE_ES_MemPoolStats_t.

12.54.1 Detailed Description

Name Memory Pool Statistics Packet

Definition at line 1467 of file cfe_es_msg.h.

12.54.2 Field Documentation

12.54.2.1 PoolHandle

`CFE_ES_MemHandle_t CFE_ES_PoolStatsTlm_Payload_t::PoolHandle`

Telemetry Mnemonic(s) **`$sc_$cpu_ES_PoolHandle`**

Definition at line 1469 of file cfe_es_msg.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

12.54.2.2 PoolStats

`CFE_ES_MemPoolStats_t CFE_ES_PoolStatsTlm_Payload_t::PoolStats`

Definition at line 1471 of file cfe_es_msg.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.55 CFE_ES_ReloadApp_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_AppReloadCmd_Payload_t Payload

12.55.1 Detailed Description

Definition at line 1291 of file cfe_es_msg.h.

12.55.2 Field Documentation

12.55.2.1 CmdHeader

```
uint8 CFE_ES_ReloadApp_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1293 of file cfe_es_msg.h.

12.55.2.2 Payload

```
CFE_ES_AppReloadCmd_Payload_t CFE_ES_ReloadApp_t::Payload
```

Definition at line 1294 of file cfe_es_msg.h.

Referenced by CFE_ES_ReloadAppCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.56 CFE_ES_ResetData_t Struct Reference

```
#include <cfe_es_resetdata_typedef.h>
```

Data Fields

- CFE_ES_ERLog_t ERLog [CFE_PLATFORM_ES_ER_LOG_ENTRIES]
- uint32 ERLogIndex
- uint32 ERLogEntries
- uint32 LastAppId
- char SystemLog [CFE_PLATFORM_ES_SYSTEM_LOG_SIZE]
- size_t SystemLogWriteIdx
- size_t SystemLogEndIdx
- uint32 SystemLogMode
- uint32 SystemLogEntryNum
- CFE_ES_PerfData_t Perf
- CFE_ES_ResetVariables_t ResetVars
- CFE_TIME_ResetVars_t TimeResetVars

12.56.1 Detailed Description

Definition at line 61 of file cfe_es_resetdata_typedef.h.

12.56.2 Field Documentation

12.56.2.1 ERLog

```
CFE_ES_ERLog_t CFE_ES_ResetData_t::ERLog[CFE_PLATFORM_ES_ER_LOG_ENTRIES]
```

Definition at line 66 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_ClearERLogCmd(), and CFE_ES_WriteToERLog().

12.56.2.2 ERLogEntries

```
uint32 CFE_ES_ResetData_t::ERLogEntries
```

Definition at line 68 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_ClearERLogCmd(), CFE_ES_HousekeepingCmd(), and CFE_ES_WriteToERLog().

12.56.2.3 ERLogIndex

```
uint32 CFE_ES_ResetData_t::ERLogIndex
```

Definition at line 67 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_ClearERLogCmd(), CFE_ES_HousekeepingCmd(), and CFE_ES_WriteToERLog().

12.56.2.4 LastAppId

```
uint32 CFE_ES_ResetData_t::LastAppId
```

Definition at line 69 of file cfe_es_resetdata_typedef.h.

12.56.2.5 Perf

```
CFE_ES_PerfData_t CFE_ES_ResetData_t::Perf
```

Definition at line 83 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), and CFE_ES_SetupPerfVariables().

12.56.2.6 ResetVars

```
CFE_ES_ResetVariables_t CFE_ES_ResetData_t::ResetVars
```

Definition at line 88 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_GetResetType(), CFE_ES_HousekeepingCmd(), CFE_ES_ProcessCoreException(), CFE_↩ ES_ResetCFE(), CFE_ES_ResetPRCountCmd(), CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetupResetVariables(), and CFE_ES_WriteToERLog().

12.56.2.7 SystemLog

```
char CFE_ES_ResetData_t::SystemLog[CFE_PLATFORM_ES_SYSTEM_LOG_SIZE]
```

Definition at line 74 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogReadData(), and CFE_ES_SysLogReadStart_↩ Unsync().

12.56.2.8 SystemLogEndIdx

```
size_t CFE_ES_ResetData_t::SystemLogEndIdx
```

Definition at line 76 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogClear_Unsync(), and CFE_ES_SysLogReadStart_Unsync().

12.56.2.9 SystemLogEntryNum

```
uint32 CFE_ES_ResetData_t::SystemLogEntryNum
```

Definition at line 78 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_SysLogAppend_Unsync(), and CFE_ES_SysLogClear_↩ Unsync().

12.56.2.10 SystemLogMode

```
uint32 CFE_ES_ResetData_t::SystemLogMode
```

Definition at line 77 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogSetMode(), and CFE_ES_TaskInit().

12.56.2.11 SystemLogWriteIdx

```
size_t CFE_ES_ResetData_t::SystemLogWriteIdx
```

Definition at line 75 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogClear_Unsync(), and CFE_ES_SysLogRead↩ Start_Unsync().

12.56.2.12 TimeResetVars

```
CFE_TIME_ResetVars_t CFE_ES_ResetData_t::TimeResetVars
```

Definition at line 94 of file cfe_es_resetdata_typedef.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_resetdata_typedef.h

12.57 CFE_ES_ResetVariables_t Struct Reference

```
#include <cfe_es_resetdata_typedef.h>
```

Data Fields

- uint32 ResetType
- uint32 ResetSubtype
- uint32 BootSource
- uint32 ES_CausedReset
- uint32 ProcessorResetCount
- uint32 MaxProcessorResetCount

12.57.1 Detailed Description

Definition at line 45 of file cfe_es_resetdata_typedef.h.

12.57.2 Field Documentation

12.57.2.1 BootSource

```
uint32 CFE_ES_ResetVariables_t::BootSource
```

Definition at line 49 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_SetupResetVariables(), and CFE_ES_WriteToERLog().

12.57.2.2 ES_CausedReset

```
uint32 CFE_ES_ResetVariables_t::ES_CausedReset
```

Definition at line 50 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), and CFE_ES_SetupResetVariables().

12.57.2.3 MaxProcessorResetCount

```
uint32 CFE_ES_ResetVariables_t::MaxProcessorResetCount
```

Definition at line 52 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), CFE_ES_↩SetMaxPRCountCmd(), CFE_ES_SetupResetVariables(), and CFE_ES_WriteToERLog().

12.57.2.4 ProcessorResetCount

```
uint32 CFE_ES_ResetVariables_t::ProcessorResetCount
```

Definition at line 51 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), CFE_ES_↩ResetPRCountCmd(), CFE_ES_SetupResetVariables(), and CFE_ES_WriteToERLog().

12.57.2.5 ResetSubtype

```
uint32 CFE_ES_ResetVariables_t::ResetSubtype
```

Definition at line 48 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_GetResetType(), CFE_ES_HousekeepingCmd(), and CFE_ES_SetupResetVariables().

12.57.2.6 ResetType

```
uint32 CFE_ES_ResetVariables_t::ResetType
```

Definition at line 47 of file cfe_es_resetdata_typedef.h.

Referenced by CFE_ES_GetResetType(), CFE_ES_HousekeepingCmd(), and CFE_ES_SetupResetVariables().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_es_resetdata_typedef.h

12.58 CFE_ES_Restart_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_RestartCmd_Payload_t Payload

12.58.1 Detailed Description

Definition at line 1149 of file cfe_es_msg.h.

12.58.2 Field Documentation

12.58.2.1 CmdHeader

```
uint8 CFE_ES_Restart_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1151 of file cfe_es_msg.h.

12.58.2.2 Payload

CFE_ES_RestartCmd_Payload_t CFE_ES_Restart_t::Payload

Definition at line 1152 of file cfe_es_msg.h.

Referenced by CFE_ES_RestartCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.59 CFE_ES_RestartCmd_Payload_t Struct Reference

Restart cFE Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- uint16 RestartType
  CFE_PSP_RST_TYPE_PROCESSOR=Processor Reset or *CFE_PSP_RST_TYPE_POWERON*=Power-On Reset

12.59.1 Detailed Description

For command details, see CFE_ES_RESTART_CC

Definition at line 1143 of file cfe_es_msg.h.

12.59.2 Field Documentation

12.59.2.1 RestartType

uint16 CFE_ES_RestartCmd_Payload_t::RestartType

Definition at line 1145 of file cfe_es_msg.h.

Referenced by CFE_ES_RestartCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.60 CFE_ES_SendMemPoolStats_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

  cFE Software Bus Command Message Header
- CFE_ES_SendMemPoolStatsCmd_Payload_t Payload

12.60.1 Detailed Description

Definition at line 1420 of file cfe_es_msg.h.

12.60.2 Field Documentation

12.60.2.1 CmdHeader

`uint8 CFE_ES_SendMemPoolStats_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 1422 of file cfe_es_msg.h.

12.60.2.2 Payload

`CFE_ES_SendMemPoolStatsCmd_Payload_t CFE_ES_SendMemPoolStats_t::Payload`

Definition at line 1423 of file cfe_es_msg.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.61 CFE_ES_SendMemPoolStatsCmd_Payload_t Struct Reference

Telemeter Memory Pool Statistics Command.

`#include <cfe_es_msg.h>`

Data Fields

- char Application [CFE_MISSION_MAX_API_LEN]
  - *RESERVED - should be all zeroes*
- CFE_ES_MemHandle_t PoolHandle

  Handle of Pool whose statistics are to be telemetered.

12.61.1 Detailed Description

For command details, see CFE_ES_SEND_MEM_POOL_STATS_CC

Definition at line 1413 of file cfe_es_msg.h.

12.61.2 Field Documentation

12.61.2.1 Application

```
char CFE_ES_SendMemPoolStatsCmd_Payload_t::Application[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1415 of file cfe_es_msg.h.

12.61.2.2 PoolHandle

```
CFE_ES_MemHandle_t CFE_ES_SendMemPoolStatsCmd_Payload_t::PoolHandle
```

Definition at line 1416 of file cfe_es_msg.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.62 CFE_ES_SetMaxPRCount_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

  cFE Software Bus Command Message Header
- CFE_ES_SetMaxPRCountCmd_Payload_t Payload

12.62.1 Detailed Description

Definition at line 1309 of file cfe_es_msg.h.

12.62.2 Field Documentation

12.62.2.1 CmdHeader

```
uint8 CFE_ES_SetMaxPRCount_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1311 of file cfe_es_msg.h.

12.62.2.2 Payload

```
CFE_ES_SetMaxPRCountCmd_Payload_t CFE_ES_SetMaxPRCount_t::Payload
```

Definition at line 1312 of file cfe_es_msg.h.

Referenced by CFE_ES_SetMaxPRCountCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.63 CFE_ES_SetMaxPRCountCmd_Payload_t Struct Reference

Set Maximum Processor Reset Count Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- uint16 MaxPRCount

  New maximum number of Processor Resets before an automatic Power-On Reset is performed.

12.63.1 Detailed Description

For command details, see CFE_ES_SET_MAX_PR_COUNT_CC

Definition at line 1303 of file cfe_es_msg.h.

12.63.2 Field Documentation

12.63.2.1 MaxPRCount

```
uint16 CFE_ES_SetMaxPRCountCmd_Payload_t::MaxPRCount
```

Definition at line 1305 of file cfe_es_msg.h.

Referenced by CFE_ES_SetMaxPRCountCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.64 CFE_ES_SetPerfFilterMask_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_SetPerfFilterMaskCmd_Payload_t Payload

12.64.1 Detailed Description

Definition at line 1382 of file cfe_es_msg.h.

12.64.2 Field Documentation

12.64.2.1 CmdHeader

```
uint8 CFE_ES_SetPerfFilterMask_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1384 of file cfe_es_msg.h.

12.64.2.2 Payload

CFE_ES_SetPerfFilterMaskCmd_Payload_t CFE_ES_SetPerfFilterMask_t::Payload

Definition at line 1385 of file cfe_es_msg.h.

Referenced by CFE_ES_SetPerfFilterMaskCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.65 CFE_ES_SetPerfFilterMaskCmd_Payload_t Struct Reference

Set Performance Analyzer Filter Mask Command.

#include <cfe_es_msg.h>

Data Fields

- uint32 FilterMaskNum

  Index into array of Filter Masks.
- uint32 FilterMask

  New Mask for specified entry in array of Filter Masks.

12.65.1 Detailed Description

For command details, see CFE_ES_SET_PERF_FILTER_MASK_CC

Definition at line 1375 of file cfe_es_msg.h.

12.65.2 Field Documentation

12.65.2.1 FilterMask

uint32 CFE_ES_SetPerfFilterMaskCmd_Payload_t::FilterMask

Definition at line 1378 of file cfe_es_msg.h.

Referenced by CFE_ES_SetPerfFilterMaskCmd().

12.65.2.2 FilterMaskNum

```
uint32 CFE_ES_SetPerfFilterMaskCmd_Payload_t::FilterMaskNum
```

Definition at line 1377 of file cfe_es_msg.h.

Referenced by CFE_ES_SetPerfFilterMaskCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.66 CFE_ES_SetPerfTriggerMask_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_SetPerfTrigMaskCmd_Payload_t Payload

12.66.1 Detailed Description

Definition at line 1401 of file cfe_es_msg.h.

12.66.2 Field Documentation

12.66.2.1 CmdHeader

```
uint8 CFE_ES_SetPerfTriggerMask_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1403 of file cfe_es_msg.h.

12.66.2.2 Payload

```
CFE_ES_SetPerfTrigMaskCmd_Payload_t CFE_ES_SetPerfTriggerMask_t::Payload
```

Definition at line 1404 of file cfe_es_msg.h.

Referenced by CFE_ES_SetPerfTriggerMaskCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.67 CFE_ES_SetPerfTrigMaskCmd_Payload_t Struct Reference

Set Performance Analyzer Trigger Mask Command.

`#include <cfe_es_msg.h>`

Data Fields

- uint32 TriggerMaskNum

  Index into array of Trigger Masks.
- uint32 TriggerMask

  New Mask for specified entry in array of Trigger Masks.

12.67.1 Detailed Description

For command details, see CFE_ES_SET_PERF_TRIGGER_MASK_CC

Definition at line 1394 of file cfe_es_msg.h.

12.67.2 Field Documentation

12.67.2.1 TriggerMask

`uint32 CFE_ES_SetPerfTrigMaskCmd_Payload_t::TriggerMask`

Definition at line 1397 of file cfe_es_msg.h.

Referenced by CFE_ES_SetPerfTriggerMaskCmd().

12.67.2.2 TriggerMaskNum

`uint32 CFE_ES_SetPerfTrigMaskCmd_Payload_t::TriggerMaskNum`

Definition at line 1396 of file cfe_es_msg.h.

Referenced by CFE_ES_SetPerfTriggerMaskCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.68 CFE_ES_Shell_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_ShellCmd_Payload_t Payload

12.68.1 Detailed Description

Definition at line 1169 of file cfe_es_msg.h.

12.68.2 Field Documentation

12.68.2.1 CmdHeader

`uint8 CFE_ES_Shell_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 1171 of file cfe_es_msg.h.

12.68.2.2 Payload

`CFE_ES_ShellCmd_Payload_t CFE_ES_Shell_t::Payload`

Definition at line 1172 of file cfe_es_msg.h.

Referenced by CFE_ES_ShellCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.69 CFE_ES_ShellCmd_Payload_t Struct Reference

Shell Command.

`#include <cfe_es_msg.h>`

Data Fields

- char CmdString [CFE_MISSION_ES_MAX_SHELL_CMD]

  ASCII text string containing shell command to be executed.
- char OutputFilename [CFE_MISSION_MAX_PATH_LEN]

  Filename where shell command output is to be written.

12.69.1 Detailed Description

For command details, see CFE_ES_SHELL_CC

Definition at line 1161 of file cfe_es_msg.h.

12.69.2 Field Documentation

12.69.2.1 CmdString

```
char CFE_ES_ShellCmd_Payload_t::CmdString[CFE_MISSION_ES_MAX_SHELL_CMD]
```

Definition at line 1163 of file cfe_es_msg.h.

Referenced by CFE_ES_ShellCmd().

12.69.2.2 OutputFilename

```
char CFE_ES_ShellCmd_Payload_t::OutputFilename[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 1165 of file cfe_es_msg.h.

Referenced by CFE_ES_ShellCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.70 CFE_ES_ShellPacket_Payload_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- char ShellOutput [CFE_MISSION_ES_MAX_SHELL_PKT]
  ASCII text string containing output from OS Shell that was received in response to an OS Shell Command.

12.70.1 Detailed Description

Name OS Shell Output Packet

Definition at line 1581 of file cfe_es_msg.h.

12.70.2 Field Documentation

12.70.2.1 ShellOutput

```
char CFE_ES_ShellPacket_Payload_t::ShellOutput[CFE_MISSION_ES_MAX_SHELL_PKT]
```

Definition at line 1583 of file cfe_es_msg.h.

Referenced by CFE_ES_ShellOutputCommand().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.71 CFE_ES_ShellTlm_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
  cFE Software Bus Telemetry Message Header
- CFE_ES_ShellPacket_Payload_t Payload

12.71.1 Detailed Description

Definition at line 1587 of file cfe_es_msg.h.

12.71.2 Field Documentation

12.71.2.1 Payload

```
CFE_ES_ShellPacket_Payload_t CFE_ES_ShellTlm_t::Payload
```

Definition at line 1590 of file cfe_es_msg.h.

Referenced by CFE_ES_ShellOutputCommand().

12.71.2.2 TlmHeader

```
uint8 CFE_ES_ShellTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1589 of file cfe_es_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.72 CFE_ES_StartApp_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_StartAppCmd_Payload_t Payload

12.72.1 Detailed Description

Definition at line 1246 of file cfe_es_msg.h.

12.72.2 Field Documentation

12.72.2.1 CmdHeader

```
uint8 CFE_ES_StartApp_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1248 of file cfe_es_msg.h.

12.72.2.2 Payload

```
CFE_ES_StartAppCmd_Payload_t CFE_ES_StartApp_t::Payload
```

Definition at line 1249 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.73 CFE_ES_StartAppCmd_Payload_t Struct Reference

Start Application Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- char Application [CFE_MISSION_MAX_API_LEN]

  Name of Application to be started.
- char AppEntryPoint [CFE_MISSION_MAX_API_LEN]

  Symbolic name of Application's entry point.
- char AppFileName [CFE_MISSION_MAX_PATH_LEN]

  Full path and filename of Application's executable image.
- uint32 StackSize

  Desired stack size for the new application.
- uint16 ExceptionAction

  CFE_ES_ExceptionAction_RESTART_APP=On exception, restart Application, CFE_ES_ExceptionAction_PROC_RE↩START=On exception, perform a Processor Reset
- uint16 Priority

  The new Applications runtime priority.

12.73.1 Detailed Description

For command details, see CFE_ES_START_APP_CC

Definition at line 1229 of file cfe_es_msg.h.

12.73.2 Field Documentation

12.73.2.1 AppEntryPoint

```
char CFE_ES_StartAppCmd_Payload_t::AppEntryPoint[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1232 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

12.73.2.2 AppFileName

```
char CFE_ES_StartAppCmd_Payload_t::AppFileName[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 1233 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

12.73.2.3 Application

```
char CFE_ES_StartAppCmd_Payload_t::Application[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1231 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

12.73.2.4 ExceptionAction

```
uint16 CFE_ES_StartAppCmd_Payload_t::ExceptionAction
```

Definition at line 1238 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

12.73.2.5 Priority

```
uint16 CFE_ES_StartAppCmd_Payload_t::Priority
```

Definition at line 1242 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

12.73.2.6 StackSize

```
uint32 CFE_ES_StartAppCmd_Payload_t::StackSize
```

Definition at line 1236 of file cfe_es_msg.h.

Referenced by CFE_ES_StartAppCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.74 CFE_ES_StartPerfCmd_Payload_t Struct Reference

Start Performance Analyzer Command.

```
#include <cfe_es_msg.h>
```

Data Fields

- uint32 TriggerMode
  Desired trigger position (Start, Center, End)

12.74.1 Detailed Description

For command details, see CFE_ES_START_PERF_DATA_CC

Definition at line 1339 of file cfe_es_msg.h.

12.74.2 Field Documentation

12.74.2.1 TriggerMode

```
uint32 CFE_ES_StartPerfCmd_Payload_t::TriggerMode
```

Definition at line 1341 of file cfe_es_msg.h.

Referenced by CFE_ES_StartPerfDataCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.75 CFE_ES_StartPerfData_t Struct Reference

`#include <cfe_es_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_ES_StartPerfCmd_Payload_t Payload

12.75.1 Detailed Description

Definition at line 1344 of file cfe_es_msg.h.

12.75.2 Field Documentation

12.75.2.1 CmdHeader

`uint8 CFE_ES_StartPerfData_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 1346 of file cfe_es_msg.h.

12.75.2.2 Payload

`CFE_ES_StartPerfCmd_Payload_t CFE_ES_StartPerfData_t::Payload`

Definition at line 1347 of file cfe_es_msg.h.

Referenced by CFE_ES_StartPerfDataCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.76 CFE_ES_StopPerfCmd_Payload_t Struct Reference

Stop Performance Analyzer Command.

`#include <cfe_es_msg.h>`

Data Fields

- char DataFileName [CFE_MISSION_MAX_PATH_LEN]

  ASCII text string of full path and filename of file Performance Analyzer data is to be written.

12.76.1 Detailed Description

For command details, see CFE_ES_STOP_PERF_DATA_CC

Definition at line 1356 of file cfe_es_msg.h.

12.76.2 Field Documentation

12.76.2.1 DataFileName

```
char CFE_ES_StopPerfCmd_Payload_t::DataFileName[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 1358 of file cfe_es_msg.h.

Referenced by CFE_ES_StopPerfDataCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.77 CFE_ES_StopPerfData_t Struct Reference

```
#include <cfe_es_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

  cFE Software Bus Command Message Header
- CFE_ES_StopPerfCmd_Payload_t Payload

12.77.1 Detailed Description

Definition at line 1362 of file cfe_es_msg.h.

12.77.2 Field Documentation

12.77.2.1 CmdHeader

```
uint8 CFE_ES_StopPerfData_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1364 of file cfe_es_msg.h.

12.77.2.2 Payload

```
CFE_ES_StopPerfCmd_Payload_t CFE_ES_StopPerfData_t::Payload
```

Definition at line 1365 of file cfe_es_msg.h.

Referenced by CFE_ES_StopPerfDataCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es_msg.h

12.78 CFE_ES_SysLogReadBuffer_t Struct Reference

Buffer structure for reading data out of the Syslog.

```
#include <cfe_es_log.h>
```

Data Fields

- size_t SizeLeft
- size_t BlockSize
- size_t EndIdx
- size_t LastOffset
- char Data [CFE_ES_SYSLOG_READ_BUFFER_SIZE]

12.78.1 Detailed Description

Access to the syslog must be synchronized, so it is not possible to directly access the contents. This structure keeps the state of read operations such that the syslog can be read in segments.

See also

CFE_ES_SysLogReadData(), CFE_ES_SysLogReadStart_Unsync()

Definition at line 124 of file cfe_es_log.h.

12.78.2 Field Documentation

12.78.2.1 BlockSize

```
size_t CFE_ES_SysLogReadBuffer_t::BlockSize
```

Size of content currently in the "Data" member

Definition at line 127 of file cfe_es_log.h.

Referenced by CFE_ES_SysLogReadData(), and CFE_ES_SysLogReadStart_Unsync().

12.78.2.2 Data

```
char CFE_ES_SysLogReadBuffer_t::Data[CFE_ES_SYSLOG_READ_BUFFER_SIZE]
```

Actual syslog content

Definition at line 131 of file cfe_es_log.h.

Referenced by CFE_ES_SysLogReadData().

12.78.2.3 EndIdx

```
size_t CFE_ES_SysLogReadBuffer_t::EndIdx
```

End of the syslog buffer at the time reading started

Definition at line 128 of file cfe_es_log.h.

Referenced by CFE_ES_SysLogReadData(), and CFE_ES_SysLogReadStart_Unsync().

12.78.2.4 LastOffset

```
size_t CFE_ES_SysLogReadBuffer_t::LastOffset
```

Current Read Position

Definition at line 129 of file cfe_es_log.h.

Referenced by CFE_ES_SysLogReadData(), and CFE_ES_SysLogReadStart_Unsync().

12.78.2.5 SizeLeft

```
size_t CFE_ES_SysLogReadBuffer_t::SizeLeft
```

Total amount of unread syslog data

Definition at line 126 of file cfe_es_log.h.

Referenced by CFE_ES_SysLogReadData(), and CFE_ES_SysLogReadStart_Unsync().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_log.h

12.79 CFE_ES_TaskData_t Struct Reference

```
#include <cfe_es_task.h>
```

Data Fields

- uint8 CommandCounter
- uint8 CommandErrorCounter
- CFE_ES_HousekeepingTlm_t HkPacket
- CFE_ES_ShellTlm_t ShellPacket
- CFE_ES_OneAppTlm_t OneAppPacket
- CFE_ES_MemStatsTlm_t MemStatsPacket
- CFE_SB_MsgPtr_t MsgPtr
- CFE_SB_PipeId_t CmdPipe
- char PipeName [OS_MAX_API_NAME]
- uint16 PipeDepth
- uint8 LimitHK
- uint8 LimitCmd

12.79.1 Detailed Description

Definition at line 66 of file cfe_es_task.h.

12.79.2 Field Documentation

12.79.2.1 CmdPipe

```
CFE_SB_PipeId_t CFE_ES_TaskData_t::CmdPipe
```

Definition at line 99 of file cfe_es_task.h.

Referenced by CFE_ES_TaskInit(), and CFE_ES_TaskMain().

12.79.2.2 CommandCounter

```
uint8 CFE_ES_TaskData_t::CommandCounter
```

Definition at line 71 of file cfe_es_task.h.

Referenced by CFE_ES_ClearERLogCmd(), CFE_ES_ClearSyslogCmd(), CFE_ES_DeleteCDSCmd(), CFE_ES_↩ DumpCDSRegistryCmd(), CFE_ES_HousekeepingCmd(), CFE_ES_NoopCmd(), CFE_ES_OverWriteSyslogCmd(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_ResetCountersCmd(), CFE_ES_ResetPRCountCmd(), CFE_ES_RestartAppCmd(), CFE_ES_SendMem↩ PoolStatsCmd(), CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTrigger↩ MaskCmd(), CFE_ES_ShellCmd(), CFE_ES_StartAppCmd(), CFE_ES_StartPerfDataCmd(), CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_TaskInit(), CFE_ES_WriteERLogCmd(), and CFE_ES_WriteSyslogCmd().

12.79.2.3 CommandErrorCounter

```
uint8 CFE_ES_TaskData_t::CommandErrorCounter
```

Definition at line 72 of file cfe_es_task.h.

Referenced by CFE_ES_DeleteCDSCmd(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_HousekeepingCmd(), CF↩ E_ES_OverWriteSyslogCmd(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_ResetCountersCmd(), CFE_ES_RestartAppCmd(), CFE_ES_RestartCmd(), CFE_ES_SendMemPoolStatsCmd(), CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_↩ ES_ShellCmd(), CFE_ES_StartAppCmd(), CFE_ES_StartPerfDataCmd(), CFE_ES_StopAppCmd(), CFE_ES_Stop↩ PerfDataCmd(), CFE_ES_TaskInit(), CFE_ES_TaskPipe(), CFE_ES_VerifyCmdLength(), CFE_ES_WriteERLogCmd(), and CFE_ES_WriteSyslogCmd().

12.79.2.4 HkPacket

```
CFE_ES_HousekeepingTlm_t CFE_ES_TaskData_t::HkPacket
```

Definition at line 77 of file cfe_es_task.h.

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_SysLogDump(), and CFE_ES_TaskInit().

12.79.2.5 LimitCmd

```
uint8 CFE_ES_TaskData_t::LimitCmd
```

Definition at line 108 of file cfe_es_task.h.

Referenced by CFE_ES_TaskInit().

12.79.2.6 LimitHK

```
uint8 CFE_ES_TaskData_t::LimitHK
```

Definition at line 107 of file cfe_es_task.h.

Referenced by CFE_ES_TaskInit().

12.79.2.7 MemStatsPacket

```
CFE_ES_MemStatsTlm_t CFE_ES_TaskData_t::MemStatsPacket
```

Definition at line 93 of file cfe_es_task.h.

Referenced by CFE_ES_SendMemPoolStatsCmd(), and CFE_ES_TaskInit().

12.79.2.8 MsgPtr

```
CFE_SB_MsgPtr_t CFE_ES_TaskData_t::MsgPtr
```

Definition at line 98 of file cfe_es_task.h.

Referenced by CFE_ES_TaskMain().

12.79.2.9 OneAppPacket

```
CFE_ES_OneAppTlm_t CFE_ES_TaskData_t::OneAppPacket
```

Definition at line 88 of file cfe_es_task.h.

Referenced by CFE_ES_QueryOneCmd(), and CFE_ES_TaskInit().

12.79.2.10 PipeDepth

```
uint16 CFE_ES_TaskData_t::PipeDepth
```

Definition at line 105 of file cfe_es_task.h.

Referenced by CFE_ES_TaskInit().

12.79.2.11 PipeName

```
char CFE_ES_TaskData_t::PipeName[OS_MAX_API_NAME]
```

Definition at line 104 of file cfe_es_task.h.

Referenced by CFE_ES_TaskInit().

12.79.2.12 ShellPacket

```
CFE_ES_ShellTlm_t CFE_ES_TaskData_t::ShellPacket
```

Definition at line 83 of file cfe_es_task.h.

Referenced by CFE_ES_ShellOutputCommand(), and CFE_ES_TaskInit().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_task.h

12.80 CFE_ES_TaskInfo_t Struct Reference

```
#include <cfe_es.h>
```

Data Fields

- uint32 TaskId

  Task Id.
- uint32 ExecutionCounter
- uint8 TaskName [OS_MAX_API_NAME]

  Task Execution Counter.
- uint32 AppId

  Parent Application ID.
- uint8 AppName [OS_MAX_API_NAME]

  Parent Application Name.

12.80.1 Detailed Description

Definition at line 258 of file cfe_es.h.

12.80.2 Field Documentation

12.80.2.1 AppId

```
uint32 CFE_ES_TaskInfo_t::AppId
```

Definition at line 263 of file cfe_es.h.

Referenced by CFE_ES_GetTaskInfo(), CFE_ES_ListTasks(), and CFE_ES_ProcessCoreException().

12.80.2.2 AppName

```
uint8 CFE_ES_TaskInfo_t::AppName[OS_MAX_API_NAME]
```

Definition at line 264 of file cfe_es.h.

Referenced by CFE_ES_GetTaskInfo(), CFE_ES_ListTasks(), and CFE_SB_GetAppTskName().

12.80.2.3 ExecutionCounter

```
uint32 CFE_ES_TaskInfo_t::ExecutionCounter
```

Definition at line 261 of file cfe_es.h.

Referenced by CFE_ES_GetTaskInfo().

12.80.2.4 TaskId

```
uint32 CFE_ES_TaskInfo_t::TaskId
```

Definition at line 260 of file cfe_es.h.

Referenced by CFE_ES_GetTaskInfo(), and CFE_ES_ListTasks().

12.80.2.5 TaskName

```
uint8 CFE_ES_TaskInfo_t::TaskName[OS_MAX_API_NAME]
```

KTask Name

Definition at line 262 of file cfe_es.h.

Referenced by CFE_ES_GetTaskInfo(), CFE_ES_ListTasks(), and CFE_SB_GetAppTskName().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_es.h

12.81 CFE_ES_TaskRecord_t Struct Reference

```
#include <cfe_es_apps.h>
```

Data Fields

- bool RecordUsed
- uint32 AppId
- uint32 TaskId
- uint32 ExecutionCounter
- char TaskName [OS_MAX_API_NAME]

12.81.1 Detailed Description

Definition at line 117 of file cfe_es_apps.h.

12.81.2 Field Documentation

12.81.2.1 AppId

```
uint32 CFE_ES_TaskRecord_t::AppId
```

Definition at line 120 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateChildTask(), CFE_ES_Create↩Objects(), CFE_ES_GetAppIDInternal(), CFE_ES_GetAppInfoInternal(), and CFE_ES_GetTaskInfo().

12.81.2.2 ExecutionCounter

```
uint32 CFE_ES_TaskRecord_t::ExecutionCounter
```

Definition at line 122 of file cfe_es_apps.h.

Referenced by CFE_ES_GetAppInfoInternal(), CFE_ES_GetTaskInfo(), CFE_ES_IncrementTaskCounter(), and CFE_ES_RunLoop().

12.81.2.3 RecordUsed

```
bool CFE_ES_TaskRecord_t::RecordUsed
```

Definition at line 119 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CleanupTaskResources(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), CFE_ES_GetAppIDInternal(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetTaskInfo(), CFE_ES_ListTasks(), CFE_ES_Main(), CFE_ES_ProcessCoreException(), and CFE_ES_QueryAllTasksCmd().

12.81.2.4 TaskId

```
uint32 CFE_ES_TaskRecord_t::TaskId
```

Definition at line 121 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_GetAppInfoInternal(), CFE_ES_ListTasks(), CFE_ES_ProcessCoreException(), and CFE_ES_QueryAllTasksCmd().

12.81.2.5 TaskName

```
char CFE_ES_TaskRecord_t::TaskName[OS_MAX_API_NAME]
```

Definition at line 123 of file cfe_es_apps.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), and CFE_ES_GetTaskInfo().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_es_apps.h

12.82 CFE_EVS_AppDataCmd_Payload_t Struct Reference

Write Event Services Application Information to File Command.

`#include <cfe_evs_msg.h>`

Data Fields

- char AppDataFilename [CFE_MISSION_MAX_PATH_LEN]

  Filename where applicaton data is to be written.

12.82.1 Detailed Description

For command details, see CFE_EVS_WRITE_APP_DATA_FILE_CC

Definition at line 955 of file cfe_evs_msg.h.

12.82.2 Field Documentation

12.82.2.1 AppDataFilename

`char CFE_EVS_AppDataCmd_Payload_t::AppDataFilename[CFE_MISSION_MAX_PATH_LEN]`

Definition at line 956 of file cfe_evs_msg.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.83 CFE_EVS_AppDataFile_t Struct Reference

`#include <cfe_evs_task.h>`

Data Fields

- char AppName [OS_MAX_API_NAME]
- uint8 ActiveFlag
- uint8 EventTypesActiveFlag
- uint16 EventCount
- EVS_BinFilter_t Filters [CFE_PLATFORM_EVS_MAX_EVENT_FILTERS]

12.83.1 Detailed Description

Definition at line 102 of file cfe_evs_task.h.

12.83.2 Field Documentation

12.83.2.1 ActiveFlag

```
uint8 CFE_EVS_AppDataFile_t::ActiveFlag
```

Definition at line 104 of file cfe_evs_task.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

12.83.2.2 AppName

```
char CFE_EVS_AppDataFile_t::AppName[OS_MAX_API_NAME]
```

Definition at line 103 of file cfe_evs_task.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

12.83.2.3 EventCount

```
uint16 CFE_EVS_AppDataFile_t::EventCount
```

Definition at line 106 of file cfe_evs_task.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

12.83.2.4 EventTypesActiveFlag

```
uint8 CFE_EVS_AppDataFile_t::EventTypesActiveFlag
```

Definition at line 105 of file cfe_evs_task.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

12.83.2.5 Filters

```
EVS_BinFilter_t CFE_EVS_AppDataFile_t::Filters[CFE_PLATFORM_EVS_MAX_EVENT_FILTERS]
```

Definition at line 107 of file cfe_evs_task.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/evs/cfe_evs_task.h

12.84 CFE_EVS_AppNameBitMaskCmd_Payload_t Struct Reference

Enable/Disable an Event Type for an Application.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]
  Application name to use in the command.
- uint8 BitMask
  BitMask to use in the command.
- uint8 Spare
  Pad to even byte.

12.84.1 Detailed Description

For command details, see CFE_EVS_ENABLE_APP_EVENT_TYPE_CC and/or CFE_EVS_DISABLE_APP_EVENT↩_TYPE_CC

Definition at line 1079 of file cfe_evs_msg.h.

12.84.2 Field Documentation

12.84.2.1 AppName

```
char CFE_EVS_AppNameBitMaskCmd_Payload_t::AppName[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1080 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), and CFE_EVS_EnableAppEventTypeCmd().

12.84.2.2 BitMask

```
uint8 CFE_EVS_AppNameBitMaskCmd_Payload_t::BitMask
```

Definition at line 1081 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), and CFE_EVS_EnableAppEventTypeCmd().

12.84.2.3 Spare

```
uint8 CFE_EVS_AppNameBitMaskCmd_Payload_t::Spare
```

Definition at line 1082 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.85 CFE_EVS_AppNameBitMaskCmd_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_AppNameBitMaskCmd_Payload_t Payload

12.85.1 Detailed Description

Definition at line 1085 of file cfe_evs_msg.h.

12.85.2 Field Documentation

12.85.2.1 CmdHeader

```
uint8 CFE_EVS_AppNameBitMaskCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1086 of file cfe_evs_msg.h.

12.85.2.2 Payload

CFE_EVS_AppNameBitMaskCmd_Payload_t CFE_EVS_AppNameBitMaskCmd_t::Payload

Definition at line 1087 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), and CFE_EVS_EnableAppEventTypeCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.86 CFE_EVS_AppNameCmd_Payload_t Struct Reference

Enable/Disable Application Events or Reset One or All Filter Counters.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]
  Application name to use in the command.

12.86.1 Detailed Description

For command details, see CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_RESET_APP_COUNTER_CC and/or CFE_EVS_RESET_ALL_FILTERS_CC

Definition at line 1030 of file cfe_evs_msg.h.

12.86.2 Field Documentation

12.86.2.1 AppName

char CFE_EVS_AppNameCmd_Payload_t::AppName[CFE_MISSION_MAX_API_LEN]

Definition at line 1031 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventsCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_ResetAllFilters↩Cmd(), and CFE_EVS_ResetAppCounterCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.87 CFE_EVS_AppNameCmd_t Struct Reference

`#include <cfe_evs_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_AppNameCmd_Payload_t Payload

12.87.1 Detailed Description

Definition at line 1034 of file cfe_evs_msg.h.

12.87.2 Field Documentation

12.87.2.1 CmdHeader

`uint8 CFE_EVS_AppNameCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 1035 of file cfe_evs_msg.h.

12.87.2.2 Payload

`CFE_EVS_AppNameCmd_Payload_t CFE_EVS_AppNameCmd_t::Payload`

Definition at line 1036 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventsCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_ResetAllFilters↩Cmd(), and CFE_EVS_ResetAppCounterCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.88 CFE_EVS_AppNameEventIDCmd_Payload_t Struct Reference

Reset an Event Filter for an Application.

`#include <cfe_evs_msg.h>`

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]

  Application name to use in the command.
- uint16 EventID

  Event ID to use in the command.

12.88.1 Detailed Description

For command details, see CFE_EVS_RESET_FILTER_CC

Definition at line 1055 of file cfe_evs_msg.h.

12.88.2 Field Documentation

12.88.2.1 AppName

```
char CFE_EVS_AppNameEventIDCmd_Payload_t::AppName[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1056 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DeleteEventFilterCmd(), and CFE_EVS_ResetFilterCmd().

12.88.2.2 EventID

```
uint16 CFE_EVS_AppNameEventIDCmd_Payload_t::EventID
```

Definition at line 1057 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DeleteEventFilterCmd(), and CFE_EVS_ResetFilterCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.89 CFE_EVS_AppNameEventIDCmd_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_AppNameEventIDCmd_Payload_t Payload

12.89.1 Detailed Description

Definition at line 1060 of file cfe_evs_msg.h.

12.89.2 Field Documentation

12.89.2.1 CmdHeader

```
uint8 CFE_EVS_AppNameEventIDCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1061 of file cfe_evs_msg.h.

12.89.2.2 Payload

```
CFE_EVS_AppNameEventIDCmd_Payload_t CFE_EVS_AppNameEventIDCmd_t::Payload
```

Definition at line 1062 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DeleteEventFilterCmd(), and CFE_EVS_ResetFilterCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.90 CFE_EVS_AppNameEventIDMaskCmd_Payload_t Struct Reference

Set, Add or Delete an Event Filter for an Application.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]

  Application name to use in the command.
- uint16 EventID

  Event ID to use in the command.
- uint16 Mask

  Mask to use in the command.

12.90.1 Detailed Description

For command details, see CFE_EVS_SET_FILTER_CC, CFE_EVS_ADD_EVENT_FILTER_CC and/or CFE_EVS_↩ DELETE_EVENT_FILTER_CC

Definition at line 1105 of file cfe_evs_msg.h.

12.90.2 Field Documentation

12.90.2.1 AppName

```
char CFE_EVS_AppNameEventIDMaskCmd_Payload_t::AppName[CFE_MISSION_MAX_API_LEN]
```

Definition at line 1106 of file cfe_evs_msg.h.

Referenced by CFE_EVS_AddEventFilterCmd(), and CFE_EVS_SetFilterCmd().

12.90.2.2 EventID

```
uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload_t::EventID
```

Definition at line 1107 of file cfe_evs_msg.h.

Referenced by CFE_EVS_AddEventFilterCmd(), and CFE_EVS_SetFilterCmd().

12.90.2.3 Mask

```
uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload_t::Mask
```

Definition at line 1108 of file cfe_evs_msg.h.

Referenced by CFE_EVS_AddEventFilterCmd(), and CFE_EVS_SetFilterCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.91 CFE_EVS_AppNameEventIDMaskCmd_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_AppNameEventIDMaskCmd_Payload_t Payload

12.91.1 Detailed Description

Definition at line 1111 of file cfe_evs_msg.h.

12.91.2 Field Documentation

12.91.2.1 CmdHeader

```
uint8 CFE_EVS_AppNameEventIDMaskCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1112 of file cfe_evs_msg.h.

12.91.2.2 Payload

```
CFE_EVS_AppNameEventIDMaskCmd_Payload_t CFE_EVS_AppNameEventIDMaskCmd_t::Payload
```

Definition at line 1113 of file cfe_evs_msg.h.

Referenced by CFE_EVS_AddEventFilterCmd(), and CFE_EVS_SetFilterCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.92 CFE_EVS_AppTlmData_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint32 AppID

  Numerical application identifier.
- uint16 AppMessageSentCounter

  Application message sent counter.
- uint8 AppEnableStatus

  Application event service enable status.
- uint8 Padding

  Padding for 32 bit boundary.

12.92.1 Detailed Description

Definition at line 1128 of file cfe_evs_msg.h.

12.92.2 Field Documentation

12.92.2.1 AppEnableStatus

`uint8 CFE_EVS_AppTlmData_t::AppEnableStatus`

Telemetry Mnemonic(s) **`$sc_$cpu_EVS_APP[CFE_ES_MAX_APPLICATIONS]`**.APPENASTAT

Definition at line 1133 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ReportHousekeepingCmd().

12.92.2.2 AppID

`uint32 CFE_EVS_AppTlmData_t::AppID`

Telemetry Mnemonic(s) **`$sc_$cpu_EVS_APP[CFE_ES_MAX_APPLICATIONS]`**.APPID

Definition at line 1129 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ReportHousekeepingCmd().

12.92.2.3 AppMessageSentCounter

`uint16 CFE_EVS_AppTlmData_t::AppMessageSentCounter`

Telemetry Mnemonic(s) **`$sc_$cpu_EVS_APP[CFE_ES_MAX_APPLICATIONS]`**.APPMSGSENTC

Definition at line 1131 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ReportHousekeepingCmd().

12.92.2.4 Padding

```
uint8 CFE_EVS_AppTlmData_t::Padding
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_APP[CFE_ES_MAX_APPLICATIONS]**.SPARE2ALIGN3

Definition at line 1135 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.93 CFE_EVS_BinFilter_t Struct Reference

```
#include <cfe_evs.h>
```

Data Fields

- uint16 EventID
  Numerical event identifier.
- uint16 Mask
  Binary filter mask value.

12.93.1 Detailed Description

Event message filter defintion structure

Definition at line 111 of file cfe_evs.h.

12.93.2 Field Documentation

12.93.2.1 EventID

```
uint16 CFE_EVS_BinFilter_t::EventID
```

Definition at line 112 of file cfe_evs.h.

Referenced by CFE_EVS_Register(), and CFE_SB_AppInit().

12.93.2.2 Mask

```
uint16 CFE_EVS_BinFilter_t::Mask
```

Definition at line 113 of file cfe_evs.h.

Referenced by CFE_EVS_Register(), and CFE_SB_AppInit().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs.h

12.94 CFE_EVS_BitMaskCmd_Payload_t Struct Reference

Enable/Disable Events or Ports Commands.

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 BitMask

  BitMask to use in the command.
- uint8 Spare

  Pad to even byte.

12.94.1 Detailed Description

For command details, see CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_ENABLE_PORTS_CC and/or CFE_EVS_DISABLE_PORTS_CC

Definition at line 1003 of file cfe_evs_msg.h.

12.94.2 Field Documentation

12.94.2.1 BitMask

```
uint8 CFE_EVS_BitMaskCmd_Payload_t::BitMask
```

Definition at line 1004 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableEventTypeCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_EnableEventTypeCmd(), and CFE_EVS_EnablePortsCmd().

12.94.2.2 Spare

```
uint8 CFE_EVS_BitMaskCmd_Payload_t::Spare
```

Definition at line 1005 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.95 CFE_EVS_BitMaskCmd_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_BitMaskCmd_Payload_t Payload

12.95.1 Detailed Description

Definition at line 1008 of file cfe_evs_msg.h.

12.95.2 Field Documentation

12.95.2.1 CmdHeader

```
uint8 CFE_EVS_BitMaskCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 1009 of file cfe_evs_msg.h.

12.95.2.2 Payload

```
CFE_EVS_BitMaskCmd_Payload_t CFE_EVS_BitMaskCmd_t::Payload
```

Definition at line 1010 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableEventTypeCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_EnableEventType↩Cmd(), and CFE_EVS_EnablePortsCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.96 CFE_EVS_GlobalData_t Struct Reference

`#include <cfe_evs_task.h>`

Data Fields

- EVS_AppData_t AppData [CFE_PLATFORM_ES_MAX_APPLICATIONS]
- CFE_EVS_Log_t * EVS_LogPtr
- CFE_EVS_HousekeepingTlm_t EVS_TlmPkt
- CFE_SB_PipeId_t EVS_CommandPipe
- uint32 EVS_SharedDataMutexID
- uint32 EVS_AppID

12.96.1 Detailed Description

Definition at line 113 of file cfe_evs_task.h.

12.96.2 Field Documentation

12.96.2.1 AppData

`EVS_AppData_t CFE_EVS_GlobalData_t::AppData[CFE_PLATFORM_ES_MAX_APPLICATIONS]`

Definition at line 115 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_CleanUpApp(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableEventTypeCmd(), CFE_EVS_Register(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), CFE_EVS_SetFilterCmd(), CFE_EVS_Unregister(), CFE_EVS_WriteAppDataFileCmd(), EVS_DisableTypes(), EVS_EnableTypes(), EVS_GenerateEventTelemetry(), EVS_GetApplicationInfo(), EVS_IsFiltered(), and EVS_NotRegistered().

12.96.2.2 EVS_AppID

`uint32 CFE_EVS_GlobalData_t::EVS_AppID`

Definition at line 126 of file cfe_evs_task.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_TaskInit(), and EVS_SendEvent().

12.96.2.3 EVS_CommandPipe

```
CFE_SB_PipeId_t CFE_EVS_GlobalData_t::EVS_CommandPipe
```

Definition at line 124 of file cfe_evs_task.h.

Referenced by CFE_EVS_TaskInit(), and CFE_EVS_TaskMain().

12.96.2.4 EVS_LogPtr

```
CFE_EVS_Log_t* CFE_EVS_GlobalData_t::EVS_LogPtr
```

Definition at line 117 of file cfe_evs_task.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_SetLogModeCmd(), CFE_↩EVS_WriteLogDataFileCmd(), EVS_AddLog(), and EVS_ClearLog().

12.96.2.5 EVS_SharedDataMutexID

```
uint32 CFE_EVS_GlobalData_t::EVS_SharedDataMutexID
```

Definition at line 125 of file cfe_evs_task.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_SetLogModeCmd(), CFE_EVS_WriteLogDataFileCmd(), EVS_Add↩Log(), and EVS_ClearLog().

12.96.2.6 EVS_TlmPkt

```
CFE_EVS_HousekeepingTlm_t CFE_EVS_GlobalData_t::EVS_TlmPkt
```

Definition at line 123 of file cfe_evs_task.h.

Referenced by CFE_EVS_ClearLogCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_EarlyInit(), CFE_EVS_Enable↩PortsCmd(), CFE_EVS_ProcessCommandPacket(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_Report↩HousekeepingCmd(), CFE_EVS_ResetCountersCmd(), CFE_EVS_SetEventFormatModeCmd(), CFE_EVS_Set↩LogModeCmd(), CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), EVS_GenerateEventTelemetry(), EVS_Not↩Registered(), and EVS_SendViaPorts().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/evs/cfe_evs_task.h

12.97 CFE_EVS_HousekeepingTlm_Payload_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CommandCounter

  EVS Command Counter.
- uint8 CommandErrorCounter

  EVS Command Error Counter.
- uint8 MessageFormatMode

  Event message format mode (short/long)
- uint8 MessageTruncCounter

  Event message truncation counter.
- uint8 UnregisteredAppCounter

  Unregistered application message send counter.
- uint8 OutputPort

  Output port mask.
- uint8 LogFullFlag

  Local event log full flag.
- uint8 LogMode

  Local event logging mode (overwrite/discard)
- uint16 MessageSendCounter

  Event message send counter.
- uint16 LogOverflowCounter

  Local event log overflow counter.
- uint8 LogEnabled

  Current event log enable/disable state.
- uint8 Spare1

  Padding for 32 bit boundary.
- uint8 Spare2

  Padding for 32 bit boundary.
- uint8 Spare3

  Padding for 32 bit boundary.
- CFE_EVS_AppTlmData_t AppData [CFE_MISSION_ES_MAX_APPLICATIONS]

  Array of registered application table data.

12.97.1 Detailed Description

Name Event Services Housekeeping Telemetry Packet

Definition at line 1144 of file cfe_evs_msg.h.

12.97.2 Field Documentation

12.97.2.1 AppData

```
CFE_EVS_AppTlmData_t CFE_EVS_HousekeepingTlm_Payload_t::AppData[CFE_MISSION_ES_MAX_APPLICATIONS]
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_APP[CFE_ES_MAX_APPLICATIONS]**

Definition at line 1177 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ReportHousekeepingCmd().

12.97.2.2 CommandCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::CommandCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_CMDPC**

Definition at line 1145 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand(), and CFE_EVS_ResetCountersCmd().

12.97.2.3 CommandErrorCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_CMDEC**

Definition at line 1147 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessCommandPacket(), CFE_EVS_ProcessGroundCommand(), and CFE_EVS_↩ ResetCountersCmd().

12.97.2.4 LogEnabled

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_LOGENABLED**

Definition at line 1168 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ClearLogCmd(), CFE_EVS_EarlyInit(), CFE_EVS_ReportHousekeepingCmd(), CFE_EV↩ S_SetLogModeCmd(), CFE_EVS_WriteLogDataFileCmd(), and EVS_AddLog().

12.97.2.5 LogFullFlag

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::LogFullFlag
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_LOGFULL**

Definition at line 1158 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EarlyInit(), and CFE_EVS_ReportHousekeepingCmd().

12.97.2.6 LogMode

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::LogMode
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_LOGMODE**

Definition at line 1160 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EarlyInit(), and CFE_EVS_ReportHousekeepingCmd().

12.97.2.7 LogOverflowCounter

```
uint16 CFE_EVS_HousekeepingTlm_Payload_t::LogOverflowCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_LOGOVERFLOWC**

Definition at line 1165 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ReportHousekeepingCmd().

12.97.2.8 MessageFormatMode

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::MessageFormatMode
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_MSGFMTMODE**

Definition at line 1149 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_SetEventFormatModeCmd(), and EVS_GenerateEventTelemetry().

12.97.2.9 MessageSendCounter

```
uint16 CFE_EVS_HousekeepingTlm_Payload_t::MessageSendCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_MSGSENTC**

Definition at line 1163 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ResetCountersCmd(), and EVS_GenerateEventTelemetry().

12.97.2.10 MessageTruncCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::MessageTruncCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_MSGTRUNC**

Definition at line 1151 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ResetCountersCmd(), and EVS_GenerateEventTelemetry().

12.97.2.11 OutputPort

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::OutputPort
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_OUTPUTPORT**

Definition at line 1156 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisablePortsCmd(), CFE_EVS_EarlyInit(), CFE_EVS_EnablePortsCmd(), and EVS_Send↩ ViaPorts().

12.97.2.12 Spare1

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::Spare1
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_HK_SPARE1**

Definition at line 1170 of file cfe_evs_msg.h.

12.97.2.13 Spare2

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::Spare2
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_HK_SPARE2**

Definition at line 1172 of file cfe_evs_msg.h.

12.97.2.14 Spare3

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::Spare3
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_HK_SPARE3**

Definition at line 1174 of file cfe_evs_msg.h.

12.97.2.15 UnregisteredAppCounter

```
uint8 CFE_EVS_HousekeepingTlm_Payload_t::UnregisteredAppCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_UNREGAPPC**

Definition at line 1154 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ResetCountersCmd(), and EVS_NotRegistered().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.98 CFE_EVS_HousekeepingTlm_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
- CFE_EVS_HousekeepingTlm_Payload_t Payload

12.98.1 Detailed Description

Definition at line 1182 of file cfe_evs_msg.h.

12.98.2 Field Documentation

12.98.2.1 Payload

```
CFE_EVS_HousekeepingTlm_Payload_t CFE_EVS_HousekeepingTlm_t::Payload
```

Definition at line 1184 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ClearLogCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_EarlyInit(), CFE_EVS_Enable↩ PortsCmd(), CFE_EVS_ProcessCommandPacket(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_Report↩ HousekeepingCmd(), CFE_EVS_ResetCountersCmd(), CFE_EVS_SetEventFormatModeCmd(), CFE_EVS_Set↩ LogModeCmd(), CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), EVS_GenerateEventTelemetry(), EVS_Not↩ Registered(), and EVS_SendViaPorts().

12.98.2.2 TlmHeader

```
uint8 CFE_EVS_HousekeepingTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1183 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.99 CFE_EVS_Log_t Struct Reference

```
#include <cfe_evs_log_typedef.h>
```

Data Fields

- uint16 Next
  Index of the next entry in the local event log.
- uint16 LogCount
  Local Event Kog counter.
- uint8 LogFullFlag
  Local Event Log full flag.
- uint8 LogMode
  Local Event Logging mode (overwrite/discard)
- uint16 LogOverflowCounter
  Local Event Log overflow counter.
- CFE_EVS_LongEventTlm_t LogEntry [CFE_PLATFORM_EVS_LOG_MAX]
  The actual Local Event Log entry.

12.99.1 Detailed Description

Definition at line 41 of file cfe_evs_log_typedef.h.

12.99.2 Field Documentation

12.99.2.1 LogCount

```
uint16 CFE_EVS_Log_t::LogCount
```

Definition at line 43 of file cfe_evs_log_typedef.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), and EVS_ClearLog().

12.99.2.2 LogEntry

```
CFE_EVS_LongEventTlm_t CFE_EVS_Log_t::LogEntry[CFE_PLATFORM_EVS_LOG_MAX]
```

Definition at line 47 of file cfe_evs_log_typedef.h.

Referenced by CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), and EVS_ClearLog().

12.99.2.3 LogFullFlag

```
uint8 CFE_EVS_Log_t::LogFullFlag
```

Definition at line 44 of file cfe_evs_log_typedef.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_ReportHousekeepingCmd(), EVS_AddLog(), and EVS_ClearLog().

12.99.2.4 LogMode

```
uint8 CFE_EVS_Log_t::LogMode
```

Definition at line 45 of file cfe_evs_log_typedef.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_SetLogModeCmd(), and E↩
VS_AddLog().

12.99.2.5 LogOverflowCounter

```
uint16 CFE_EVS_Log_t::LogOverflowCounter
```

Definition at line 46 of file cfe_evs_log_typedef.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_ReportHousekeepingCmd(), EVS_AddLog(), and EVS_ClearLog().

12.99.2.6 Next

```
uint16 CFE_EVS_Log_t::Next
```

Definition at line 42 of file cfe_evs_log_typedef.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), and EVS_ClearLog().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/private/cfe_evs_log_typedef.h

12.100 CFE_EVS_LogFileCmd_Payload_t Struct Reference

Write Event Log to File Command.

```
#include <cfe_evs_msg.h>
```

Data Fields

- char LogFilename [CFE_MISSION_MAX_PATH_LEN]
  Filename where log data is to be written.

12.100.1 Detailed Description

For command details, see CFE_EVS_WRITE_LOG_DATA_FILE_CC

Definition at line 939 of file cfe_evs_msg.h.

12.100.2 Field Documentation

12.100.2.1 LogFilename

```
char CFE_EVS_LogFileCmd_Payload_t::LogFilename[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 940 of file cfe_evs_msg.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.101 CFE_EVS_LongEventTlm_Payload_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- CFE_EVS_PacketID_t PacketID
  Event packet information.
- char Message [CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]
  Event message string.
- uint8 Spare1
  Structure padding.
- uint8 Spare2
  Structure padding.

12.101.1 Detailed Description

Name Event Message Telemetry Packet (Long format)

Definition at line 1207 of file cfe_evs_msg.h.

12.101.2 Field Documentation

12.101.2.1 Message

```
char CFE_EVS_LongEventTlm_Payload_t::Message[CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_EVENT[CFE_EVS_MAX_MESSAGE_LENGTH]**

Definition at line 1209 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

12.101.2.2 PacketID

```
CFE_EVS_PacketID_t CFE_EVS_LongEventTlm_Payload_t::PacketID
```

Definition at line 1208 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

12.101.2.3 Spare1

```
uint8 CFE_EVS_LongEventTlm_Payload_t::Spare1
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_SPARE1**

Definition at line 1211 of file cfe_evs_msg.h.

12.101.2.4 Spare2

```
uint8 CFE_EVS_LongEventTlm_Payload_t::Spare2
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_SPARE2**

Definition at line 1213 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.102 CFE_EVS_LongEventTlm_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
- CFE_EVS_LongEventTlm_Payload_t Payload

12.102.1 Detailed Description

Definition at line 1225 of file cfe_evs_msg.h.

12.102.2 Field Documentation

12.102.2.1 Payload

```
CFE_EVS_LongEventTlm_Payload_t CFE_EVS_LongEventTlm_t::Payload
```

Definition at line 1227 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

12.102.2.2 TlmHeader

```
uint8 CFE_EVS_LongEventTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1226 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.103 CFE_EVS_NoArgsCmd_t Struct Reference

Command with no additional arguments.

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

12.103.1 Detailed Description

Definition at line 920 of file cfe_evs_msg.h.

12.103.2 Field Documentation

12.103.2.1 CmdHeader

```
uint8 CFE_EVS_NoArgsCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 921 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.104 CFE_EVS_PacketID_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- char AppName [CFE_MISSION_MAX_API_LEN]
  Application name.
- uint16 EventID
  Numerical event identifier.
- uint16 EventType
  Numerical event type identifier.
- uint32 SpacecraftID
  Spacecraft identifier.
- uint32 ProcessorID
  Numerical processor identifier.

12.104.1 Detailed Description

Telemetry packet structures

Definition at line 1189 of file cfe_evs_msg.h.

12.104.2 Field Documentation

12.104.2.1 AppName

```
char CFE_EVS_PacketID_t::AppName[CFE_MISSION_MAX_API_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_APPNAME[OS_MAX_API_NAME]**

Definition at line 1190 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

12.104.2.2 EventID

```
uint16 CFE_EVS_PacketID_t::EventID
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_EVENTID**

Definition at line 1192 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

12.104.2.3 EventType

```
uint16 CFE_EVS_PacketID_t::EventType
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_EVENTTYPE**

Definition at line 1194 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry().

12.104.2.4 ProcessorID

```
uint32 CFE_EVS_PacketID_t::ProcessorID
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_PROCESSORID**

Definition at line 1198 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

12.104.2.5 SpacecraftID

```
uint32 CFE_EVS_PacketID_t::SpacecraftID
```

Telemetry Mnemonic(s) **$sc_$cpu_EVS_SCID**

Definition at line 1196 of file cfe_evs_msg.h.

Referenced by EVS_GenerateEventTelemetry(), and EVS_SendViaPorts().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.105 CFE_EVS_SetEventFormatMode_Payload_t Struct Reference

Set Event Format Mode or Set Log Mode Commands.

`#include <cfe_evs_msg.h>`

Data Fields

- CFE_EVS_MsgFormat_Enum_t MsgFormat
  Mode to use in the command.
- uint8 Spare
  Pad to even byte.

12.105.1 Detailed Description

For command details, see CFE_EVS_SET_EVENT_FORMAT_MODE_CC and/or CFE_EVS_SET_LOG_MODE_CC

Definition at line 986 of file cfe_evs_msg.h.

12.105.2 Field Documentation

12.105.2.1 MsgFormat

`CFE_EVS_MsgFormat_Enum_t CFE_EVS_SetEventFormatMode_Payload_t::MsgFormat`

Definition at line 987 of file cfe_evs_msg.h.

Referenced by CFE_EVS_SetEventFormatModeCmd().

12.105.2.2 Spare

`uint8 CFE_EVS_SetEventFormatMode_Payload_t::Spare`

Definition at line 988 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.106 CFE_EVS_SetEventFormatMode_t Struct Reference

`#include <cfe_evs_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_SetEventFormatMode_Payload_t Payload

12.106.1 Detailed Description

Definition at line 991 of file cfe_evs_msg.h.

12.106.2 Field Documentation

12.106.2.1 CmdHeader

```
uint8 CFE_EVS_SetEventFormatMode_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 992 of file cfe_evs_msg.h.

12.106.2.2 Payload

```
CFE_EVS_SetEventFormatMode_Payload_t CFE_EVS_SetEventFormatMode_t::Payload
```

Definition at line 993 of file cfe_evs_msg.h.

Referenced by CFE_EVS_SetEventFormatModeCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.107 CFE_EVS_SetLogMode_Payload_t Struct Reference

Set Event Format Mode or Set Log Mode Commands.

```
#include <cfe_evs_msg.h>
```

Data Fields

- CFE_EVS_LogMode_Enum_t LogMode
  Mode to use in the command.
- uint8 Spare
  Pad to even byte.

12.107.1 Detailed Description

For command details, see CFE_EVS_SET_EVENT_FORMAT_MODE_CC and/or CFE_EVS_SET_LOG_MODE_CC

Definition at line 970 of file cfe_evs_msg.h.

12.107.2 Field Documentation

12.107.2.1 LogMode

```
CFE_EVS_LogMode_Enum_t CFE_EVS_SetLogMode_Payload_t::LogMode
```

Definition at line 971 of file cfe_evs_msg.h.

Referenced by CFE_EVS_SetLogModeCmd().

12.107.2.2 Spare

```
uint8 CFE_EVS_SetLogMode_Payload_t::Spare
```

Definition at line 972 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.108 CFE_EVS_SetLogMode_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_SetLogMode_Payload_t Payload

12.108.1 Detailed Description

Definition at line 975 of file cfe_evs_msg.h.

12.108.2 Field Documentation

12.108.2.1 CmdHeader

```
uint8 CFE_EVS_SetLogMode_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 976 of file cfe_evs_msg.h.

12.108.2.2 Payload

```
CFE_EVS_SetLogMode_Payload_t CFE_EVS_SetLogMode_t::Payload
```

Definition at line 977 of file cfe_evs_msg.h.

Referenced by CFE_EVS_SetLogModeCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.109 CFE_EVS_ShortEventTlm_Payload_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- CFE_EVS_PacketID_t PacketID
  Event packet information.

12.109.1 Detailed Description

Name Event Message Telemetry Packet (Short format)

Definition at line 1220 of file cfe_evs_msg.h.

12.109.2 Field Documentation

12.109.2.1 PacketID

```
CFE_EVS_PacketID_t CFE_EVS_ShortEventTlm_Payload_t::PacketID
```

Definition at line 1221 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.110 CFE_EVS_ShortEventTlm_t Struct Reference

```
#include <cfe_evs_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
- CFE_EVS_ShortEventTlm_Payload_t Payload

12.110.1 Detailed Description

Definition at line 1231 of file cfe_evs_msg.h.

12.110.2 Field Documentation

12.110.2.1 Payload

```
CFE_EVS_ShortEventTlm_Payload_t CFE_EVS_ShortEventTlm_t::Payload
```

Definition at line 1233 of file cfe_evs_msg.h.

12.110.2.2 TlmHeader

```
uint8 CFE_EVS_ShortEventTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1232 of file cfe_evs_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.111 CFE_EVS_WriteAppDataFile_t Struct Reference

`#include <cfe_evs_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_AppDataCmd_Payload_t Payload

12.111.1 Detailed Description

Definition at line 959 of file cfe_evs_msg.h.

12.111.2 Field Documentation

12.111.2.1 CmdHeader

`uint8 CFE_EVS_WriteAppDataFile_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 960 of file cfe_evs_msg.h.

12.111.2.2 Payload

`CFE_EVS_AppDataCmd_Payload_t CFE_EVS_WriteAppDataFile_t::Payload`

Definition at line 961 of file cfe_evs_msg.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.112 CFE_EVS_WriteLogDataFile_t Struct Reference

`#include <cfe_evs_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_EVS_LogFileCmd_Payload_t Payload

12.112.1 Detailed Description

Definition at line 943 of file cfe_evs_msg.h.

12.112.2 Field Documentation

12.112.2.1 CmdHeader

```
uint8 CFE_EVS_WriteLogDataFile_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 944 of file cfe_evs_msg.h.

12.112.2.2 Payload

```
CFE_EVS_LogFileCmd_Payload_t CFE_EVS_WriteLogDataFile_t::Payload
```

Definition at line 945 of file cfe_evs_msg.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h

12.113 CFE_FS_Header_t Struct Reference

Standard cFE File header structure definition.

```
#include <cfe_fs_extern_typedefs.h>
```

Data Fields

- uint32 ContentType
  Identifies the content type (='cFE1'=0x63464531)
- uint32 SubType
  Type of `ContentType`, if necessary.
- uint32 Length
  Length of primary header.
- uint32 SpacecraftID
  Spacecraft that generated the file.
- uint32 ProcessorID
  Processor that generated the file.
- uint32 ApplicationID
  Application that generated the file.
- uint32 TimeSeconds
  File creation timestamp (seconds)
- uint32 TimeSubSeconds
  File creation timestamp (sub-seconds)
- char Description [CFE_FS_HDR_DESC_MAX_LEN]
  File description.

12.113.1 Detailed Description

Definition at line 223 of file cfe_fs_extern_typedefs.h.

12.113.2 Field Documentation

12.113.2.1 ApplicationID

```
uint32 CFE_FS_Header_t::ApplicationID
```

Definition at line 232 of file cfe_fs_extern_typedefs.h.

12.113.2.2 ContentType

```
uint32 CFE_FS_Header_t::ContentType
```

Definition at line 225 of file cfe_fs_extern_typedefs.h.

12.113.2.3 Description

```
char CFE_FS_Header_t::Description[CFE_FS_HDR_DESC_MAX_LEN]
```

Definition at line 237 of file cfe_fs_extern_typedefs.h.

12.113.2.4 Length

```
uint32 CFE_FS_Header_t::Length
```

Definition at line 229 of file cfe_fs_extern_typedefs.h.

12.113.2.5 ProcessorID

```
uint32 CFE_FS_Header_t::ProcessorID
```

Definition at line 231 of file cfe_fs_extern_typedefs.h.

12.113.2.6 SpacecraftID

```
uint32 CFE_FS_Header_t::SpacecraftID
```

Definition at line 230 of file cfe_fs_extern_typedefs.h.

12.113.2.7 SubType

```
uint32 CFE_FS_Header_t::SubType
```

Standard SubType definitions can be found here

Definition at line 226 of file cfe_fs_extern_typedefs.h.

12.113.2.8 TimeSeconds

```
uint32 CFE_FS_Header_t::TimeSeconds
```

Definition at line 234 of file cfe_fs_extern_typedefs.h.

12.113.2.9 TimeSubSeconds

```
uint32 CFE_FS_Header_t::TimeSubSeconds
```

Definition at line 235 of file cfe_fs_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_fs_extern_typedefs.h

12.114 CFE_PSP_CommandData_t Struct Reference

Data Fields

- char ResetType [CFE_PSP_RESET_NAME_LENGTH]
- uint32 GotResetType
- uint32 SubType
- uint32 GotSubType
- char CpuName [CFE_PSP_CPU_NAME_LENGTH]
- uint32 GotCpuName
- uint32 CpuId
- uint32 GotCpuId
- uint32 SpacecraftId
- uint32 GotSpacecraftId

12.114.1 Detailed Description

Definition at line 87 of file cfe_psp_start.c.

12.114.2 Field Documentation

12.114.2.1 CpuId

```
uint32 CFE_PSP_CommandData_t::CpuId
```

Definition at line 98 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.2 CpuName

```
char CFE_PSP_CommandData_t::CpuName[CFE_PSP_CPU_NAME_LENGTH]
```

Definition at line 95 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.3 GotCpuId

```
uint32 CFE_PSP_CommandData_t::GotCpuId
```

Definition at line 99 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.4 GotCpuName

```
uint32 CFE_PSP_CommandData_t::GotCpuName
```

Definition at line 96 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.5 GotResetType

```
uint32 CFE_PSP_CommandData_t::GotResetType
```

Definition at line 90 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.6 GotSpacecraftId

```
uint32 CFE_PSP_CommandData_t::GotSpacecraftId
```

Definition at line 102 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.7 GotSubType

```
uint32 CFE_PSP_CommandData_t::GotSubType
```

Definition at line 93 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.8 ResetType

```
char CFE_PSP_CommandData_t::ResetType[CFE_PSP_RESET_NAME_LENGTH]
```

Definition at line 89 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.9 SpacecraftId

```
uint32 CFE_PSP_CommandData_t::SpacecraftId
```

Definition at line 101 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

12.114.2.10 SubType

```
uint32 CFE_PSP_CommandData_t::SubType
```

Definition at line 92 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

The documentation for this struct was generated from the following file:

- psp/fsw/pc-linux/src/cfe_psp_start.c

12.115 CFE_PSP_MemTable_t Struct Reference

```
#include <cfe_psp.h>
```

Data Fields

- uint32 MemoryType
- uint32 WordSize
- cpuaddr StartAddr
- uint32 Size
- uint32 Attributes

12.115.1 Detailed Description

Definition at line 152 of file cfe_psp.h.

12.115.2 Field Documentation

12.115.2.1 Attributes

```
uint32 CFE_PSP_MemTable_t::Attributes
```

Definition at line 158 of file cfe_psp.h.

12.115.2.2 MemoryType

```
uint32 CFE_PSP_MemTable_t::MemoryType
```

Definition at line 154 of file cfe_psp.h.

12.115.2.3 Size

```
uint32 CFE_PSP_MemTable_t::Size
```

Definition at line 157 of file cfe_psp.h.

12.115.2.4 StartAddr

```
cpuaddr CFE_PSP_MemTable_t::StartAddr
```

Definition at line 156 of file cfe_psp.h.

12.115.2.5 WordSize

```
uint32 CFE_PSP_MemTable_t::WordSize
```

Definition at line 155 of file cfe_psp.h.

The documentation for this struct was generated from the following file:

- psp/fsw/inc/cfe_psp.h

12.116 CFE_PSP_VersionInfo_t Struct Reference

```
#include <cfe_psp_configdata.h>
```

Data Fields

- uint8 MajorVersion
- uint8 MinorVersion
- uint8 Revision
- uint8 MissionRev

12.116.1 Detailed Description

Definition at line 40 of file cfe_psp_configdata.h.

12.116.2 Field Documentation

12.116.2.1 MajorVersion

```
uint8 CFE_PSP_VersionInfo_t::MajorVersion
```

Definition at line 42 of file cfe_psp_configdata.h.

12.116.2.2 MinorVersion

```
uint8 CFE_PSP_VersionInfo_t::MinorVersion
```

Definition at line 43 of file cfe_psp_configdata.h.

12.116.2.3 MissionRev

```
uint8 CFE_PSP_VersionInfo_t::MissionRev
```

Definition at line 45 of file cfe_psp_configdata.h.

12.116.2.4 Revision

```
uint8 CFE_PSP_VersionInfo_t::Revision
```

Definition at line 44 of file cfe_psp_configdata.h.

The documentation for this struct was generated from the following file:

- psp/fsw/inc/cfe_psp_configdata.h

12.117 CFE_SB_AllSubscriptionsTlm_Payload_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- uint32 PktSegment

  Pkt number(starts at 1) in the series.
- uint32 TotalSegments

  Total number of pkts needed to complete the request.
- uint32 Entries

  Number of entries in the pkt.
- CFE_SB_SubEntries_t Entry [CFE_SB_SUB_ENTRIES_PER_PKT]

  Array of CFE_SB_SubEntries_t entries.

12.117.1 Detailed Description

Name SB Previous Subscriptions Packet

This structure defines the pkt(s) sent by SB that contains a list of all current subscriptions. This pkt is generated on cmd and intended to be used primarily by the Software Bus Networking Application (SBN). Typically, when the cmd is received there are more subscriptions than can fit in one pkt. The complete list of subscriptions is sent via a series of segmented pkts.

Definition at line 740 of file cfe_sb_msg.h.

12.117.2 Field Documentation

12.117.2.1 Entries

```
uint32 CFE_SB_AllSubscriptionsTlm_Payload_t::Entries
```

Definition at line 744 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendPrevSubsCmd().

12.117.2.2 Entry

```
CFE_SB_SubEntries_t CFE_SB_AllSubscriptionsTlm_Payload_t::Entry[CFE_SB_SUB_ENTRIES_PER_PKT]
```

Definition at line 745 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendPrevSubsCmd().

12.117.2.3 PktSegment

```
uint32 CFE_SB_AllSubscriptionsTlm_Payload_t::PktSegment
```

Definition at line 742 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendPrevSubsCmd().

12.117.2.4 TotalSegments

```
uint32 CFE_SB_AllSubscriptionsTlm_Payload_t::TotalSegments
```

Definition at line 743 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.118 CFE_SB_AllSubscriptionsTlm_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_TlmHdr_t Hdr
  cFE Software Bus Telemetry Message Header
- CFE_SB_AllSubscriptionsTlm_Payload_t Payload

12.118.1 Detailed Description

Definition at line 748 of file cfe_sb_msg.h.

12.118.2 Field Documentation

12.118.2.1 Hdr

```
CFE_SB_TlmHdr_t CFE_SB_AllSubscriptionsTlm_t::Hdr
```

Definition at line 749 of file cfe_sb_msg.h.

12.118.2.2 Payload

```
CFE_SB_AllSubscriptionsTlm_Payload_t CFE_SB_AllSubscriptionsTlm_t::Payload
```

Definition at line 750 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendPrevSubsCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.119 CFE_SB_BufferD_t Struct Reference

`#include <cfe_sb_priv.h>`

Data Fields

- CFE_SB_MsgId_t MsgId
- uint16 UseCount
- uint32 Size
- void * Buffer
- CFE_SB_SenderId_t Sender

12.119.1 Detailed Description

Definition at line 173 of file cfe_sb_priv.h.

12.119.2 Field Documentation

12.119.2.1 Buffer

`void* CFE_SB_BufferD_t::Buffer`

Definition at line 177 of file cfe_sb_priv.h.

Referenced by CFE_SB_RcvMsg(), and CFE_SB_SendMsgFull().

12.119.2.2 MsgId

`CFE_SB_MsgId_t CFE_SB_BufferD_t::MsgId`

Definition at line 174 of file cfe_sb_priv.h.

Referenced by CFE_SB_GetBufferFromCaller(), and CFE_SB_RcvMsg().

12.119.2.3 Sender

`CFE_SB_SenderId_t CFE_SB_BufferD_t::Sender`

Definition at line 178 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

12.119.2.4 Size

```
uint32 CFE_SB_BufferD_t::Size
```

Definition at line 176 of file cfe_sb_priv.h.

12.119.2.5 UseCount

```
uint16 CFE_SB_BufferD_t::UseCount
```

Definition at line 175 of file cfe_sb_priv.h.

Referenced by CFE_SB_DecrBufUseCnt().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.120 CFE_SB_DestinationD_t Struct Reference

```
#include <cfe_sb_priv.h>
```

Data Fields

- CFE_SB_PipeId_t PipeId
- uint8 Active
- uint16 MsgId2PipeLim
- uint16 BuffCount
- uint16 DestCnt
- uint8 Scope
- uint8 Spare [3]
- void * Prev
- void * Next

12.120.1 Detailed Description

Definition at line 193 of file cfe_sb_priv.h.

12.120.2 Field Documentation

12.120.2.1 Active

```
uint8 CFE_SB_DestinationD_t::Active
```

Definition at line 195 of file cfe_sb_priv.h.

Referenced by CFE_SB_DisableRouteCmd(), and CFE_SB_EnableRouteCmd().

12.120.2.2 BuffCount

```
uint16 CFE_SB_DestinationD_t::BuffCount
```

Definition at line 197 of file cfe_sb_priv.h.

Referenced by CFE_SB_RcvMsg().

12.120.2.3 DestCnt

```
uint16 CFE_SB_DestinationD_t::DestCnt
```

Definition at line 198 of file cfe_sb_priv.h.

12.120.2.4 MsgId2PipeLim

```
uint16 CFE_SB_DestinationD_t::MsgId2PipeLim
```

Definition at line 196 of file cfe_sb_priv.h.

12.120.2.5 Next

```
void* CFE_SB_DestinationD_t::Next
```

Definition at line 202 of file cfe_sb_priv.h.

Referenced by CFE_SB_AddDest(), CFE_SB_DuplicateSubscribeCheck(), CFE_SB_GetDestPtr(), CFE_SB_↩ RemoveDest(), CFE_SB_SendRtgInfo(), and CFE_SB_UnsubscribeFull().

12.120.2.6 PipeId

```
CFE_SB_PipeId_t CFE_SB_DestinationD_t::PipeId
```

Definition at line 194 of file cfe_sb_priv.h.

Referenced by CFE_SB_UnsubscribeFull().

12.120.2.7 Prev

```
void* CFE_SB_DestinationD_t::Prev
```

Definition at line 201 of file cfe_sb_priv.h.

Referenced by CFE_SB_AddDest(), and CFE_SB_RemoveDest().

12.120.2.8 Scope

```
uint8 CFE_SB_DestinationD_t::Scope
```

Definition at line 199 of file cfe_sb_priv.h.

Referenced by CFE_SB_FindGlobalMsgIdCnt(), and CFE_SB_SendPrevSubsCmd().

12.120.2.9 Spare

```
uint8 CFE_SB_DestinationD_t::Spare[3]
```

Definition at line 200 of file cfe_sb_priv.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.121 CFE_SB_EventBuf_t Struct Reference

```
#include <cfe_sb_priv.h>
```

Data Fields

- uint32 EvtsToSnd
- CFE_SB_SendErrEventBuf_t EvtBuf [CFE_PLATFORM_SB_MAX_DEST_PER_PKT]

12.121.1 Detailed Description

Definition at line 330 of file cfe_sb_priv.h.

12.121.2 Field Documentation

12.121.2.1 EvtBuf

```
CFE_SB_SendErrEventBuf_t CFE_SB_EventBuf_t::EvtBuf[CFE_PLATFORM_SB_MAX_DEST_PER_PKT]
```

Definition at line 332 of file cfe_sb_priv.h.

12.121.2.2 EvtsToSnd

```
uint32 CFE_SB_EventBuf_t::EvtsToSnd
```

Definition at line 331 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.122 CFE_SB_HousekeepingTlm_Payload_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- uint8 CommandCounter

  Count of valid commands received.
- uint8 CommandErrorCounter

  Count of invalid commands received.
- uint8 NoSubscribersCounter

  Count pkts sent with no subscribers.
- uint8 MsgSendErrorCounter

  Count of message send errors.
- uint8 MsgReceiveErrorCounter

  Count of message receive errors.
- uint8 InternalErrorCounter

  Count of queue read or write errors.
- uint8 CreatePipeErrorCounter

  Count of errors in create pipe API.
- uint8 SubscribeErrorCounter

  Count of errors in subscribe API.
- uint8 PipeOptsErrorCounter

  Count of errors in set/get pipe options API.
- uint8 DuplicateSubscriptionsCounter

  Count of duplicate subscriptions.
- uint8 GetPipeIdByNameErrorCounter

  Count of errors in get pipe id by name API.
- uint8 Spare2Align [1]

  Spare bytes to ensure alignment.
- uint16 PipeOverflowErrorCounter

  Count of pipe overflow errors.
- uint16 MsgLimitErrorCounter

  Count of msg id to pipe errors.
- CFE_ES_MemHandle_t MemPoolHandle

  Handle to SB's Memory Pool.
- uint32 MemInUse

  Memory in use.
- uint32 UnmarkedMem

  cfg param CFE_PLATFORM_SB_BUF_MEMORY_BYTES minus Peak Memory in use

12.122.1 Detailed Description

Name Software Bus task housekeeping Packet

Definition at line 541 of file cfe_sb_msg.h.

12.122.2 Field Documentation

12.122.2.1 CommandCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::CommandCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_CMDPC**

Definition at line 543 of file cfe_sb_msg.h.

Referenced by CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_IncrCmdCtr(), CFE_SB_Noop↩ Cmd(), CFE_SB_ResetCounters(), and CFE_SB_SendStatsCmd().

12.122.2.2 CommandErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_CMDEC**

Definition at line 545 of file cfe_sb_msg.h.

Referenced by CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_IncrCmdCtr(), CFE_SB_↩ ProcessCmdPipePkt(), CFE_SB_ResetCounters(), and CFE_SB_VerifyCmdLength().

12.122.2.3 CreatePipeErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_NewPipeEC**

Definition at line 556 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), and CFE_SB_ResetCounters().

12.122.2.4 DuplicateSubscriptionsCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::DuplicateSubscriptionsCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_DupSubCnt**

Definition at line 562 of file cfe_sb_msg.h.

Referenced by CFE_SB_ResetCounters(), and CFE_SB_SubscribeFull().

12.122.2.5 GetPipeIdByNameErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::GetPipeIdByNameErrorCounter
```

Telemetry Mnemonic(s)

Definition at line 564 of file cfe_sb_msg.h.

Referenced by CFE_SB_GetPipeIdByName().

12.122.2.6 InternalErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::InternalErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_InternalEC**

Definition at line 554 of file cfe_sb_msg.h.

Referenced by CFE_SB_ReadQueue(), CFE_SB_ResetCounters(), and CFE_SB_SendMsgFull().

12.122.2.7 MemInUse

```
uint32 CFE_SB_HousekeepingTlm_Payload_t::MemInUse
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_MemInUse**

Definition at line 577 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendHKTlmCmd().

12.122.2.8 MemPoolHandle

```
CFE_ES_MemHandle_t CFE_SB_HousekeepingTlm_Payload_t::MemPoolHandle
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_MemPoolHdl**

Definition at line 574 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit().

12.122.2.9 MsgLimitErrorCounter

```
uint16 CFE_SB_HousekeepingTlm_Payload_t::MsgLimitErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_MsgLimEC**

Definition at line 571 of file cfe_sb_msg.h.

Referenced by CFE_SB_ResetCounters(), and CFE_SB_SendMsgFull().

12.122.2.10 MsgReceiveErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::MsgReceiveErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_MsgRecEC**

Definition at line 552 of file cfe_sb_msg.h.

Referenced by CFE_SB_RcvMsg(), and CFE_SB_ResetCounters().

12.122.2.11 MsgSendErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::MsgSendErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_MsgSndEC**

Definition at line 549 of file cfe_sb_msg.h.

Referenced by CFE_SB_ResetCounters(), and CFE_SB_SendMsgFull().

12.122.2.12 NoSubscribersCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::NoSubscribersCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_NoSubEC**

Definition at line 547 of file cfe_sb_msg.h.

Referenced by CFE_SB_ResetCounters(), and CFE_SB_SendMsgFull().

12.122.2.13 PipeOptsErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::PipeOptsErrorCounter
```

Telemetry Mnemonic(s)

Definition at line 560 of file cfe_sb_msg.h.

Referenced by CFE_SB_GetPipeOpts(), and CFE_SB_SetPipeOpts().

12.122.2.14 PipeOverflowErrorCounter

```
uint16 CFE_SB_HousekeepingTlm_Payload_t::PipeOverflowErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_PipeOvrEC**

Definition at line 569 of file cfe_sb_msg.h.

Referenced by CFE_SB_ResetCounters(), and CFE_SB_SendMsgFull().

12.122.2.15 Spare2Align

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::Spare2Align[1]
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_Spare2Align[2]**

Definition at line 566 of file cfe_sb_msg.h.

12.122.2.16 SubscribeErrorCounter

```
uint8 CFE_SB_HousekeepingTlm_Payload_t::SubscribeErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_SubscrEC**

Definition at line 558 of file cfe_sb_msg.h.

Referenced by CFE_SB_ResetCounters(), and CFE_SB_SubscribeFull().

12.122.2.17 UnmarkedMem

```
uint32 CFE_SB_HousekeepingTlm_Payload_t::UnmarkedMem
```

Telemetry Mnemonic(s) **`$sc_$cpu_SB_UnMarkedMem`**

Definition at line 580 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendHKTlmCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.123 CFE_SB_HousekeepingTlm_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_TlmHdr_t Hdr
  cFE Software Bus Telemetry Message Header
- CFE_SB_HousekeepingTlm_Payload_t Payload

12.123.1 Detailed Description

Definition at line 584 of file cfe_sb_msg.h.

12.123.2 Field Documentation

12.123.2.1 Hdr

```
CFE_SB_TlmHdr_t CFE_SB_HousekeepingTlm_t::Hdr
```

Definition at line 585 of file cfe_sb_msg.h.

12.123.2.2 Payload

```
CFE_SB_HousekeepingTlm_Payload_t CFE_SB_HousekeepingTlm_t::Payload
```

Definition at line 586 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeOpts(), CFE_SB_IncrCmdCtr(), CFE_SB_NoopCmd(), CFE_SB_ProcessCmdPipePkt(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_ResetCounters(), CFE_SB_SendHKTlmCmd(), CFE_SB_SendMsgFull(), CFE_SB_SendStatsCmd(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_VerifyCmdLength().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.124 CFE_SB_MemParams_t Struct Reference

```
#include <cfe_sb_priv.h>
```

Public Member Functions

- CFE_ES_STATIC_POOL_TYPE (CFE_PLATFORM_SB_BUF_MEMORY_BYTES) Partition

Data Fields

- CFE_ES_MemHandle_t PoolHdl

12.124.1 Detailed Description

Definition at line 272 of file cfe_sb_priv.h.

12.124.2 Member Function Documentation

12.124.2.1 CFE_ES_STATIC_POOL_TYPE()

```
CFE_SB_MemParams_t::CFE_ES_STATIC_POOL_TYPE (
            CFE_PLATFORM_SB_BUF_MEMORY_BYTES  )
```

12.124.3 Field Documentation

12.124.3.1 PoolHdl

```
CFE_ES_MemHandle_t CFE_SB_MemParams_t::PoolHdl
```

Definition at line 274 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_Init↩ Buffers(), CFE_SB_PutDestinationBlk(), CFE_SB_ReturnBufferToPool(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_↩ ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.125 CFE_SB_Msg_t Union Reference

Generic Software Bus Message Type Definition.

```
#include <cfe_sb.h>
```

Data Fields

- CCSDS_PriHdr_t Hdr

  CCSDS Primary Header CCSDS_PriHdr_t.
- CCSDS_SpacePacket_t SpacePacket
- uint32 Dword

  Forces minimum of 32-bit alignment for this object.
- uint8 Byte [sizeof(CCSDS_PriHdr_t)]

  Allows byte-level access.

12.125.1 Detailed Description

<

Definition at line 91 of file cfe_sb.h.

12.125.2 Field Documentation

12.125.2.1 Byte

```
uint8 CFE_SB_Msg_t::Byte[sizeof(CCSDS_PriHdr_t)]
```

Definition at line 95 of file cfe_sb.h.

12.125.2.2 Dword

```
uint32 CFE_SB_Msg_t::Dword
```

Definition at line 94 of file cfe_sb.h.

12.125.2.3 Hdr

```
CCSDS_PriHdr_t CFE_SB_Msg_t::Hdr
```

Definition at line 92 of file cfe_sb.h.

Referenced by CFE_SB_GenerateChecksum(), CFE_SB_GetChecksum(), CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(), CFE_SB_GetMsgTime(), CFE_SB_GetTotalMsgLength(), CFE_SB_SetCmdCode(), CFE_SB_SetMsgId(), CFE_SB_SetMsgSeqCnt(), CFE_SB_SetMsgTime(), CFE_SB_SetTotalMsgLength(), CFE_SB_SetUserDataLength(), and CFE_SB_ValidateChecksum().

12.125.2.4 SpacePacket

```
CCSDS_SpacePacket_t CFE_SB_Msg_t::SpacePacket
```

Definition at line 93 of file cfe_sb.h.

Referenced by CFE_SB_GetMsgId(), and CFE_SB_SetMsgId().

The documentation for this union was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb.h

12.126 CFE_SB_MsgKey_t Struct Reference

```
#include <cfe_sb_priv.h>
```

Data Fields

- CFE_SB_MsgKey_Atom_t KeyIdx

12.126.1 Detailed Description

Definition at line 139 of file cfe_sb_priv.h.

12.126.2 Field Documentation

12.126.2.1 KeyIdx

```
CFE_SB_MsgKey_Atom_t CFE_SB_MsgKey_t::KeyIdx
```

Holding value, do not use directly

Definition at line 141 of file cfe_sb_priv.h.

Referenced by CFE_SB_IsValidMsgKey(), and CFE_SB_MsgKeyToValue().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.127 CFE_SB_MsgMapFileEntry_t Struct Reference

SB Map File Entry.

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_MsgId_Atom_t MsgId
  Message Id which has been subscribed to.
- CFE_SB_MsgRouteIdx_Atom_t Index
  Routing table index where pipe destinations are found.

12.127.1 Detailed Description

Structure of one element of the map information in response to CFE_SB_SEND_MAP_INFO_CC

Definition at line 683 of file cfe_sb_msg.h.

12.127.2 Field Documentation

12.127.2.1 Index

```
CFE_SB_MsgRouteIdx_Atom_t CFE_SB_MsgMapFileEntry_t::Index
```

Definition at line 685 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendMapInfo().

12.127.2.2 MsgId

```
CFE_SB_MsgId_Atom_t CFE_SB_MsgMapFileEntry_t::MsgId
```

Definition at line 684 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendMapInfo().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.128 CFE_SB_MsgRouteIdx_t Struct Reference

An wrapper for holding a routing table index.

```
#include <cfe_sb_priv.h>
```

Data Fields

- CFE_SB_MsgRouteIdx_Atom_t RouteIdx

12.128.1 Detailed Description

This is intended as a form of "strong typedef" where direct assignments should be restricted. Software bus uses numeric indexes into multiple tables to perform its duties, and it is important that these index values are distinct and separate and not mixed together.

Using this holding structure prevents assignment directly into a different index or direct usage as numeric value.

Definition at line 156 of file cfe_sb_priv.h.

12.128.2 Field Documentation

12.128.2.1 RouteIdx

`CFE_SB_MsgRouteIdx_Atom_t CFE_SB_MsgRouteIdx_t::RouteIdx`

Holding value, do not use directly in code

Definition at line 158 of file cfe_sb_priv.h.

Referenced by CFE_SB_IsValidRouteIdx(), and CFE_SB_RouteIdxToValue().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.129 CFE_SB_PipeD_t Struct Reference

`#include <cfe_sb_priv.h>`

Data Fields

- uint8 InUse
- CFE_SB_PipeId_t PipeId
- char AppName [OS_MAX_API_NAME]
- uint8 Opts
- uint8 Spare
- uint32 AppId
- uint32 SysQueueId
- uint32 LastSender
- uint16 QueueDepth
- uint16 SendErrors
- CFE_SB_BufferD_t * CurrentBuff
- CFE_SB_BufferD_t * ToTrashBuff

12.129.1 Detailed Description

Definition at line 249 of file cfe_sb_priv.h.

12.129.2 Field Documentation

12.129.2.1 AppId

```
uint32 CFE_SB_PipeD_t::AppId
```

Definition at line 255 of file cfe_sb_priv.h.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLast↩SenderId(), CFE_SB_SendMsgFull(), CFE_SB_SendRtgInfo(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

12.129.2.2 AppName

```
char CFE_SB_PipeD_t::AppName[OS_MAX_API_NAME]
```

Definition at line 252 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe().

12.129.2.3 CurrentBuff

```
CFE_SB_BufferD_t* CFE_SB_PipeD_t::CurrentBuff
```

Definition at line 260 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_InitPipe↩Tbl(), and CFE_SB_RcvMsg().

12.129.2.4 InUse

```
uint8 CFE_SB_PipeD_t::InUse
```

Definition at line 250 of file cfe_sb_priv.h.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetAvailPipe↩Idx(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeIdx(), CFE_SB_InitPipeTbl(), CFE_SB_SendPipeInfo(), and CFE_SB_ValidatePipeId().

12.129.2.5 LastSender

```
uint32 CFE_SB_PipeD_t::LastSender
```

Definition at line 257 of file cfe_sb_priv.h.

12.129.2.6 Opts

```
uint8 CFE_SB_PipeD_t::Opts
```

Definition at line 253 of file cfe_sb_priv.h.

Referenced by CFE_SB_GetPipeOpts(), CFE_SB_SendMsgFull(), and CFE_SB_SetPipeOpts().

12.129.2.7 PipeId

```
CFE_SB_PipeId_t CFE_SB_PipeD_t::PipeId
```

Definition at line 251 of file cfe_sb_priv.h.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetPipeIdBy↩ Name(), CFE_SB_GetPipeIdx(), CFE_SB_InitPipeTbl(), CFE_SB_RcvMsg(), and CFE_SB_ReadQueue().

12.129.2.8 QueueDepth

```
uint16 CFE_SB_PipeD_t::QueueDepth
```

Definition at line 258 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe().

12.129.2.9 SendErrors

```
uint16 CFE_SB_PipeD_t::SendErrors
```

Definition at line 259 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe(), and CFE_SB_SendMsgFull().

12.129.2.10 Spare

```
uint8 CFE_SB_PipeD_t::Spare
```

Definition at line 254 of file cfe_sb_priv.h.

12.129.2.11 SysQueueId

```
uint32 CFE_SB_PipeD_t::SysQueueId
```

Definition at line 256 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_InitPipeTbl(), CFE_SB_ReadQueue(), and CFE_SB_SendMsgFull().

12.129.2.12 ToTrashBuff

```
CFE_SB_BufferD_t* CFE_SB_PipeD_t::ToTrashBuff
```

Definition at line 261 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), and CFE_SB_RcvMsg().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.130 CFE_SB_PipeDepthStats_t Struct Reference

SB Pipe Depth Statistics.

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_PipeId_t PipeId
  Pipe Id associated with the stats below.
- uint8 Spare
  Spare byte to ensure alignment.
- uint16 Depth
  Number of messages the pipe can hold.
- uint16 InUse
  Number of messages currently on the pipe.
- uint16 PeakInUse
  Peak number of messages that have been on the pipe.

12.130.1 Detailed Description

Used in SB Statistics Telemetry Packet CFE_SB_StatsTlm_t

Definition at line 595 of file cfe_sb_msg.h.

12.130.2 Field Documentation

12.130.2.1 Depth

`uint16 CFE_SB_PipeDepthStats_t::Depth`

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]**.SB_PDDEPTH

Definition at line 601 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), and CFE_SB_DeletePipeFull().

12.130.2.2 InUse

`uint16 CFE_SB_PipeDepthStats_t::InUse`

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]**.SB_PDINUSE

Definition at line 603 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_RcvMsg(), and CFE_SB_SendMsgFull().

12.130.2.3 PeakInUse

`uint16 CFE_SB_PipeDepthStats_t::PeakInUse`

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]**.SB_PDPKINUSE

Definition at line 605 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), and CFE_SB_SendMsgFull().

12.130.2.4 PipeId

`CFE_SB_PipeId_t CFE_SB_PipeDepthStats_t::PipeId`

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]**.SB_PDPIPEID

Definition at line 597 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), and CFE_SB_DeletePipeFull().

12.130.2.5 Spare

```
uint8 CFE_SB_PipeDepthStats_t::Spare
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]**.SB_PDSPARE

Definition at line 599 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.131 CFE_SB_Qos_t Struct Reference

```
#include <cfe_sb.h>
```

Data Fields

- uint8 Priority

  Specify high(1) or low(0) message priority for off-board routing, currently unused.
- uint8 Reliability

  Specify high(1) or low(0) message transfer reliability for off-board routing, currently unused.

12.131.1 Detailed Description

Definition at line 140 of file cfe_sb.h.

12.131.2 Field Documentation

12.131.2.1 Priority

```
uint8 CFE_SB_Qos_t::Priority
```

Definition at line 141 of file cfe_sb.h.

Referenced by CFE_SB_EarlyInit(), CFE_SB_SendPrevSubsCmd(), and CFE_SB_SubscribeFull().

12.131.2.2 Reliability

```
uint8 CFE_SB_Qos_t::Reliability
```

Definition at line 142 of file cfe_sb.h.

Referenced by CFE_SB_EarlyInit(), CFE_SB_SendPrevSubsCmd(), and CFE_SB_SubscribeFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb.h

12.132 CFE_SB_RouteCmd_Payload_t Struct Reference

Enable/Disable Route Commands.

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_MsgId_t MsgId
  Message ID of route to be enabled or disabled CFE_SB_MsgId_t.
- CFE_SB_PipeId_t Pipe
  Pipe ID of route to be enabled or disabled CFE_SB_PipeId_t.
- uint8 Spare
  Spare byte to make command even number of bytes.

12.132.1 Detailed Description

This structure contains a definition used by two SB commands, 'Enable Route' CFE_SB_ENABLE_ROUTE_CC and 'Disable Route' CFE_SB_DISABLE_ROUTE_CC. A route is the destination pipe for a particular message and is therefore defined as a MsgId and PipeId combination.

Definition at line 516 of file cfe_sb_msg.h.

12.132.2 Field Documentation

12.132.2.1 MsgId

```
CFE_SB_MsgId_t CFE_SB_RouteCmd_Payload_t::MsgId
```

Definition at line 518 of file cfe_sb_msg.h.

Referenced by CFE_SB_DisableRouteCmd(), and CFE_SB_EnableRouteCmd().

12.132.2.2 Pipe

```
CFE_SB_PipeId_t CFE_SB_RouteCmd_Payload_t::Pipe
```

Definition at line 519 of file cfe_sb_msg.h.

Referenced by CFE_SB_DisableRouteCmd(), and CFE_SB_EnableRouteCmd().

12.132.2.3 Spare

```
uint8 CFE_SB_RouteCmd_Payload_t::Spare
```

Definition at line 520 of file cfe_sb_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.133 CFE_SB_RouteCmd_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_CmdHdr_t Hdr

  cFE Software Bus Command Message Header CFE_SB_CmdHdr_t
- CFE_SB_RouteCmd_Payload_t Payload

12.133.1 Detailed Description

Definition at line 523 of file cfe_sb_msg.h.

12.133.2 Field Documentation

12.133.2.1 Hdr

```
CFE_SB_CmdHdr_t CFE_SB_RouteCmd_t::Hdr
```

Definition at line 524 of file cfe_sb_msg.h.

12.133.2.2 Payload

```
CFE_SB_RouteCmd_Payload_t CFE_SB_RouteCmd_t::Payload
```

Definition at line 525 of file cfe_sb_msg.h.

Referenced by CFE_SB_DisableRouteCmd(), and CFE_SB_EnableRouteCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.134 CFE_SB_RouteEntry_t Struct Reference

```
#include <cfe_sb_priv.h>
```

Data Fields

- CFE_SB_MsgId_t MsgId
- uint16 Destinations
- uint32 SeqCnt
- CFE_SB_DestinationD_t * ListHeadPtr

12.134.1 Detailed Description

Definition at line 233 of file cfe_sb_priv.h.

12.134.2 Field Documentation

12.134.2.1 Destinations

```
uint16 CFE_SB_RouteEntry_t::Destinations
```

Definition at line 235 of file cfe_sb_priv.h.

Referenced by CFE_SB_InitRoutingTbl(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

12.134.2.2 ListHeadPtr

```
CFE_SB_DestinationD_t* CFE_SB_RouteEntry_t::ListHeadPtr
```

Definition at line 237 of file cfe_sb_priv.h.

Referenced by CFE_SB_AddDest(), CFE_SB_DeletePipeFull(), CFE_SB_DuplicateSubscribeCheck(), CFE_SB_↩ FindGlobalMsgIdCnt(), CFE_SB_GetDestPtr(), CFE_SB_InitRoutingTbl(), CFE_SB_RemoveDest(), CFE_SB_Send↩ PrevSubsCmd(), CFE_SB_SendRtgInfo(), and CFE_SB_UnsubscribeFull().

12.134.2.3 MsgId

```
CFE_SB_MsgId_t CFE_SB_RouteEntry_t::MsgId
```

Original Message Id when the subscription was created

Definition at line 234 of file cfe_sb_priv.h.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_InitRoutingTbl(), CFE_SB_↩ SendMapInfo(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRtgInfo(), and CFE_SB_↩ SubscribeFull().

12.134.2.4 SeqCnt

```
uint32 CFE_SB_RouteEntry_t::SeqCnt
```

Definition at line 236 of file cfe_sb_priv.h.

Referenced by CFE_SB_InitRoutingTbl(), and CFE_SB_SendMsgFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.135 CFE_SB_RoutingFileEntry_t Struct Reference

SB Routing File Entry.

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_MsgId_t MsgId

  Message Id portion of the route.
- CFE_SB_PipeId_t PipeId

  Pipe Id portion of the route.
- uint8 State

  Route Enabled or Disabled.
- uint16 MsgCnt

  Number of msgs with this MsgId sent to this PipeId.
- char AppName [CFE_MISSION_MAX_API_LEN]

  Pipe Depth Statistics.
- char PipeName [CFE_MISSION_MAX_API_LEN]

  Pipe Depth Statistics.

12.135.1 Detailed Description

Structure of one element of the routing information in response to CFE_SB_SEND_ROUTING_INFO_CC

Definition at line 668 of file cfe_sb_msg.h.

12.135.2 Field Documentation

12.135.2.1 AppName

```
char CFE_SB_RoutingFileEntry_t::AppName[CFE_MISSION_MAX_API_LEN]
```

Definition at line 673 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendRtgInfo().

12.135.2.2 MsgCnt

```
uint16 CFE_SB_RoutingFileEntry_t::MsgCnt
```

Definition at line 672 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendRtgInfo().

12.135.2.3 MsgId

```
CFE_SB_MsgId_t CFE_SB_RoutingFileEntry_t::MsgId
```

Definition at line 669 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendRtgInfo().

12.135.2.4 PipeId

```
CFE_SB_PipeId_t CFE_SB_RoutingFileEntry_t::PipeId
```

Definition at line 670 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendRtgInfo().

12.135.2.5 PipeName

```
char CFE_SB_RoutingFileEntry_t::PipeName[CFE_MISSION_MAX_API_LEN]
```

Definition at line 674 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendRtgInfo().

12.135.2.6 State

```
uint8 CFE_SB_RoutingFileEntry_t::State
```

Definition at line 671 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendRtgInfo().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.136 CFE_SB_SenderId_t Struct Reference

```
#include <cfe_sb.h>
```

Data Fields

- uint32 ProcessorId

  Processor Id from which the message was sent.
- char AppName [OS_MAX_API_NAME]

  Application that sent the message.

12.136.1 Detailed Description

Definition at line 153 of file cfe_sb.h.

12.136.2 Field Documentation

12.136.2.1 AppName

```
char CFE_SB_SenderId_t::AppName[OS_MAX_API_NAME]
```

Definition at line 155 of file cfe_sb.h.

Referenced by CFE_SB_SendMsgFull().

12.136.2.2 ProcessorId

```
uint32 CFE_SB_SenderId_t::ProcessorId
```

Definition at line 154 of file cfe_sb.h.

Referenced by CFE_SB_SendMsgFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb.h

12.137 CFE_SB_SendErrEventBuf_t Struct Reference

```
#include <cfe_sb_priv.h>
```

Data Fields

- uint32 EventId
- int32 ErrStat
- CFE_SB_PipeId_t PipeId

12.137.1 Detailed Description

Definition at line 317 of file cfe_sb_priv.h.

12.137.2 Field Documentation

12.137.2.1 ErrStat

```
int32 CFE_SB_SendErrEventBuf_t::ErrStat
```

Definition at line 319 of file cfe_sb_priv.h.

12.137.2.2 EventId

```
uint32 CFE_SB_SendErrEventBuf_t::EventId
```

Definition at line 318 of file cfe_sb_priv.h.

12.137.2.3 PipeId

```
CFE_SB_PipeId_t CFE_SB_SendErrEventBuf_t::PipeId
```

Definition at line 320 of file cfe_sb_priv.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.138 CFE_SB_SingleSubscriptionTlm_Payload_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- uint8 SubType
  Subscription or Unsubscription.
- CFE_SB_MsgId_t MsgId
  MsgId subscribed or unsubscribe to.
- CFE_SB_Qos_t Qos
  Quality of Service, used only for interprocessor communication.
- CFE_SB_PipeId_t Pipe
  Destination pipe id to send above msg id.

12.138.1 Detailed Description

Name SB Subscription Report Packet

This structure defines the pkt sent by SB when a subscription or a request to unsubscribe is received while subscription reporting is enabled. By default subscription reporting is disabled. This feature is intended to be used primarily by Software Bus Networking Application (SBN)

See also

CFE_SB_ENABLE_SUB_REPORTING_CC, CFE_SB_DISABLE_SUB_REPORTING_CC

Definition at line 699 of file cfe_sb_msg.h.

12.138.2 Field Documentation

12.138.2.1 MsgId

`CFE_SB_MsgId_t CFE_SB_SingleSubscriptionTlm_Payload_t::MsgId`

Definition at line 702 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

12.138.2.2 Pipe

`CFE_SB_PipeId_t CFE_SB_SingleSubscriptionTlm_Payload_t::Pipe`

Definition at line 704 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

12.138.2.3 Qos

`CFE_SB_Qos_t CFE_SB_SingleSubscriptionTlm_Payload_t::Qos`

Definition at line 703 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

12.138.2.4 SubType

```
uint8 CFE_SB_SingleSubscriptionTlm_Payload_t::SubType
```

Definition at line 701 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.139 CFE_SB_SingleSubscriptionTlm_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_TlmHdr_t Hdr
  cFE Software Bus Telemetry Message Header
- CFE_SB_SingleSubscriptionTlm_Payload_t Payload

12.139.1 Detailed Description

Definition at line 708 of file cfe_sb_msg.h.

12.139.2 Field Documentation

12.139.2.1 Hdr

```
CFE_SB_TlmHdr_t CFE_SB_SingleSubscriptionTlm_t::Hdr
```

Definition at line 709 of file cfe_sb_msg.h.

12.139.2.2 Payload

```
CFE_SB_SingleSubscriptionTlm_Payload_t CFE_SB_SingleSubscriptionTlm_t::Payload
```

Definition at line 710 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.140 CFE_SB_StatsTlm_Payload_t Struct Reference

`#include <cfe_sb_msg.h>`

Data Fields

- uint32 MsgIdsInUse

  Current number of MsgIds with a destination.
- uint32 PeakMsgIdsInUse

  Peak number of MsgIds with a destination.
- uint32 MaxMsgIdsAllowed

  cFE Cfg Param CFE_PLATFORM_SB_MAX_MSG_IDS
- uint32 PipesInUse

  Number of pipes currently in use.
- uint32 PeakPipesInUse

  Peak number of pipes since last reboot.
- uint32 MaxPipesAllowed

  cFE Cfg Param CFE_PLATFORM_SB_MAX_PIPES
- uint32 MemInUse

  Memory bytes currently in use for SB msg transfers.
- uint32 PeakMemInUse

  Peak memory bytes in use for SB msg transfers.
- uint32 MaxMemAllowed

  cFE Cfg Param CFE_PLATFORM_SB_BUF_MEMORY_BYTES
- uint32 SubscriptionsInUse

  Number of current subscriptions.
- uint32 PeakSubscriptionsInUse

  Peak number of subscriptions.
- uint32 MaxSubscriptionsAllowed

  product of CFE_PLATFORM_SB_MAX_MSG_IDS and CFE_PLATFORM_SB_MAX_DEST_PER_PKT
- uint32 SBBuffersInUse

  Number of SB message buffers currently in use.
- uint32 PeakSBBuffersInUse

  Max number of SB message buffers in use.
- uint32 MaxPipeDepthAllowed

  cFE Cfg Param CFE_SB_MAX_PIPE_DEPTH
- CFE_SB_PipeDepthStats_t PipeDepthStats [CFE_MISSION_SB_MAX_PIPES]

  Pipe Depth Statistics CFE_SB_PipeDepthStats_t.

12.140.1 Detailed Description

Name SB Statistics Telemetry Packet

SB Statistics packet sent (via CFE_SB_SendMsg) in response to CFE_SB_SEND_SB_STATS_CC

Definition at line 615 of file cfe_sb_msg.h.

12.140.2 Field Documentation

12.140.2.1 MaxMemAllowed

`uint32 CFE_SB_StatsTlm_Payload_t::MaxMemAllowed`

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMMBMALW`**

Definition at line 635 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit().

12.140.2.2 MaxMsgIdsAllowed

`uint32 CFE_SB_StatsTlm_Payload_t::MaxMsgIdsAllowed`

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMMMIDALW`**

Definition at line 621 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit().

12.140.2.3 MaxPipeDepthAllowed

`uint32 CFE_SB_StatsTlm_Payload_t::MaxPipeDepthAllowed`

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMMPDALW`**

Definition at line 651 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit().

12.140.2.4 MaxPipesAllowed

`uint32 CFE_SB_StatsTlm_Payload_t::MaxPipesAllowed`

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMMPALW`**

Definition at line 628 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit().

12.140.2.5 MaxSubscriptionsAllowed

```
uint32 CFE_SB_StatsTlm_Payload_t::MaxSubscriptionsAllowed
```

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMMSALW`**

Definition at line 642 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit().

12.140.2.6 MemInUse

```
uint32 CFE_SB_StatsTlm_Payload_t::MemInUse
```

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMBMIU`**

Definition at line 631 of file cfe_sb_msg.h.

Referenced by CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_PutDestinationBlk(), CFE↩_SB_ReturnBufferToPool(), CFE_SB_SendHKTlmCmd(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyRelease↩Desc(), and CFE_SB_ZeroCopyReleasePtr().

12.140.2.7 MsgIdsInUse

```
uint32 CFE_SB_StatsTlm_Payload_t::MsgIdsInUse
```

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMMIDIU`**

Definition at line 617 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

12.140.2.8 PeakMemInUse

```
uint32 CFE_SB_StatsTlm_Payload_t::PeakMemInUse
```

Telemetry Mnemonic(s) **`$sc_$cpu_SB_Stat.SB_SMPBMIU`**

Definition at line 633 of file cfe_sb_msg.h.

Referenced by CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_SendHKTlmCmd(), and CF↩E_SB_ZeroCopyGetPtr().

12.140.2.9 PeakMsgIdsInUse

uint32 CFE_SB_StatsTlm_Payload_t::PeakMsgIdsInUse

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPMIDIU**

Definition at line 619 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

12.140.2.10 PeakPipesInUse

uint32 CFE_SB_StatsTlm_Payload_t::PeakPipesInUse

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPPIU**

Definition at line 626 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe().

12.140.2.11 PeakSBBuffersInUse

uint32 CFE_SB_StatsTlm_Payload_t::PeakSBBuffersInUse

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPSBBIU**

Definition at line 648 of file cfe_sb_msg.h.

Referenced by CFE_SB_GetBufferFromPool(), and CFE_SB_ZeroCopyGetPtr().

12.140.2.12 PeakSubscriptionsInUse

uint32 CFE_SB_StatsTlm_Payload_t::PeakSubscriptionsInUse

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPSIU**

Definition at line 640 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull().

12.140.2.13 PipeDepthStats

```
CFE_SB_PipeDepthStats_t CFE_SB_StatsTlm_Payload_t::PipeDepthStats[CFE_MISSION_SB_MAX_PIPES]
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPDS[CFE_SB_MAX_PIPES]**

Definition at line 653 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_RcvMsg(), and CFE_SB_SendMsgFull().

12.140.2.14 PipesInUse

```
uint32 CFE_SB_StatsTlm_Payload_t::PipesInUse
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMPIU**

Definition at line 624 of file cfe_sb_msg.h.

Referenced by CFE_SB_CreatePipe(), and CFE_SB_DeletePipeFull().

12.140.2.15 SBBuffersInUse

```
uint32 CFE_SB_StatsTlm_Payload_t::SBBuffersInUse
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMSBBIU**

Definition at line 646 of file cfe_sb_msg.h.

Referenced by CFE_SB_GetBufferFromPool(), CFE_SB_ReturnBufferToPool(), CFE_SB_ZeroCopyGetPtr(), and CFE_SB_ZeroCopyReleasePtr().

12.140.2.16 SubscriptionsInUse

```
uint32 CFE_SB_StatsTlm_Payload_t::SubscriptionsInUse
```

Telemetry Mnemonic(s) **$sc_$cpu_SB_Stat.SB_SMSIU**

Definition at line 638 of file cfe_sb_msg.h.

Referenced by CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.141 CFE_SB_StatsTlm_t Struct Reference

`#include <cfe_sb_msg.h>`

Data Fields

- CFE_SB_TlmHdr_t Hdr
  cFE Software Bus Telemetry Message Header
- CFE_SB_StatsTlm_Payload_t Payload

12.141.1 Detailed Description

Definition at line 657 of file cfe_sb_msg.h.

12.141.2 Field Documentation

12.141.2.1 Hdr

`CFE_SB_TlmHdr_t CFE_SB_StatsTlm_t::Hdr`

Definition at line 658 of file cfe_sb_msg.h.

12.141.2.2 Payload

`CFE_SB_StatsTlm_Payload_t CFE_SB_StatsTlm_t::Payload`

Definition at line 659 of file cfe_sb_msg.h.

Referenced by CFE_SB_AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_PutDestinationBlk(), CFE_SB_RcvMsg(), CFE_SB_ReturnBufferToPool(), CFE_SB_SendHKTlmCmd(), CFE_SB_SendMsgFull(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.142 CFE_SB_SubEntries_t Struct Reference

SB Previous Subscriptions Entry.

`#include <cfe_sb_msg.h>`

Data Fields

- CFE_SB_MsgId_t MsgId
  MsgId portion of the subscription.
- CFE_SB_Qos_t Qos
  Qos portion of the subscription.
- CFE_SB_PipeId_t Pipe
  PipeId portion of the subscription.

12.142.1 Detailed Description

This structure defines an entry used in the CFE_SB_PrevSubsPkt_t Intended to be used primarily by Software Bus Networking Application (SBN)

Used in structure definition CFE_SB_AllSubscriptionsTlm_t

Definition at line 722 of file cfe_sb_msg.h.

12.142.2 Field Documentation

12.142.2.1 MsgId

```
CFE_SB_MsgId_t CFE_SB_SubEntries_t::MsgId
```

Definition at line 724 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendPrevSubsCmd().

12.142.2.2 Pipe

```
CFE_SB_PipeId_t CFE_SB_SubEntries_t::Pipe
```

Definition at line 726 of file cfe_sb_msg.h.

12.142.2.3 Qos

```
CFE_SB_Qos_t CFE_SB_SubEntries_t::Qos
```

Definition at line 725 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendPrevSubsCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.143 cfe_sb_t Struct Reference

`#include <cfe_sb_priv.h>`

Data Fields

- uint32 SharedDataMutexId
- uint32 SubscriptionReporting
- uint32 SenderReporting
- uint32 AppId
- uint32 StopRecurseFlags [CFE_PLATFORM_ES_MAX_APPLICATIONS]
- void * ZeroCopyTail
- CFE_SB_PipeD_t PipeTbl [CFE_PLATFORM_SB_MAX_PIPES]
- CFE_SB_HousekeepingTlm_t HKTlmMsg
- CFE_SB_StatsTlm_t StatTlmMsg
- CFE_SB_PipeId_t CmdPipe
- CFE_SB_Msg_t * CmdPipePktPtr
- CFE_SB_MemParams_t Mem
- CFE_SB_MsgRouteIdx_t MsgMap [CFE_SB_MAX_NUMBER_OF_MSG_KEYS]
- CFE_SB_RouteEntry_t RoutingTbl [CFE_PLATFORM_SB_MAX_MSG_IDS]
- CFE_SB_AllSubscriptionsTlm_t PrevSubMsg
- CFE_SB_SingleSubscriptionTlm_t SubRprtMsg
- CFE_EVS_BinFilter_t EventFilters [CFE_SB_MAX_CFG_FILE_EVENTS_TO_FILTER]
- uint16 RouteIdxTop
- CFE_SB_MsgRouteIdx_t RouteIdxStack [CFE_PLATFORM_SB_MAX_MSG_IDS]

12.143.1 Detailed Description

Definition at line 286 of file cfe_sb_priv.h.

12.143.2 Field Documentation

12.143.2.1 AppId

`uint32 cfe_sb_t::AppId`

Definition at line 290 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

12.143.2.2 CmdPipe

CFE_SB_PipeId_t cfe_sb_t::CmdPipe

Definition at line 296 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_TaskMain().

12.143.2.3 CmdPipePktPtr

CFE_SB_Msg_t* cfe_sb_t::CmdPipePktPtr

Definition at line 297 of file cfe_sb_priv.h.

Referenced by CFE_SB_ProcessCmdPipePkt(), and CFE_SB_TaskMain().

12.143.2.4 EventFilters

CFE_EVS_BinFilter_t cfe_sb_t::EventFilters[CFE_SB_MAX_CFG_FILE_EVENTS_TO_FILTER]

Definition at line 303 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit().

12.143.2.5 HKTlmMsg

CFE_SB_HousekeepingTlm_t cfe_sb_t::HKTlmMsg

Definition at line 294 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeOpts(), CFE_SB_IncrCmdCtr(), CFE_SB_NoopCmd(), CFE_SB_ProcessCmdPipePkt(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_ResetCounters(), CFE_SB_SendHKTlmCmd(), CFE_SB_SendMsgFull(), CFE_SB_SendStatsCmd(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_VerifyCmdLength().

12.143.2.6 Mem

CFE_SB_MemParams_t cfe_sb_t::Mem

Definition at line 298 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_InitBuffers(), CFE_SB_PutDestinationBlk(), CFE_SB_ReturnBufferToPool(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

12.143.2.7 MsgMap

```
CFE_SB_MsgRouteIdx_t cfe_sb_t::MsgMap[CFE_SB_MAX_NUMBER_OF_MSG_KEYS]
```

Definition at line 299 of file cfe_sb_priv.h.

Referenced by CFE_SB_GetRoutingTblIdx(), CFE_SB_InitMsgMap(), CFE_SB_SendRtgInfo(), and CFE_SB_Set↩ RoutingTblIdx().

12.143.2.8 PipeTbl

```
CFE_SB_PipeD_t cfe_sb_t::PipeTbl[CFE_PLATFORM_SB_MAX_PIPES]
```

Definition at line 293 of file cfe_sb_priv.h.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetAvailPipe↩ Idx(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_InitPipeTbl(), CFE_SB_SendMsgFull(), CFE_SB_Send↩ PipeInfo(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), and CFE_SB_Validate↩ PipeId().

12.143.2.9 PrevSubMsg

```
CFE_SB_AllSubscriptionsTlm_t cfe_sb_t::PrevSubMsg
```

Definition at line 301 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_SendPrevSubsCmd().

12.143.2.10 RouteIdxStack

```
CFE_SB_MsgRouteIdx_t cfe_sb_t::RouteIdxStack[CFE_PLATFORM_SB_MAX_MSG_IDS]
```

Definition at line 306 of file cfe_sb_priv.h.

Referenced by CFE_SB_InitIdxStack(), CFE_SB_RouteIdxPop_Unsync(), and CFE_SB_RouteIdxPush_Unsync().

12.143.2.11 RouteIdxTop

```
uint16 cfe_sb_t::RouteIdxTop
```

Definition at line 305 of file cfe_sb_priv.h.

Referenced by CFE_SB_InitIdxStack(), CFE_SB_RouteIdxPop_Unsync(), and CFE_SB_RouteIdxPush_Unsync().

12.143.2.12 RoutingTbl

```
CFE_SB_RouteEntry_t cfe_sb_t::RoutingTbl[CFE_PLATFORM_SB_MAX_MSG_IDS]
```

Definition at line 300 of file cfe_sb_priv.h.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_InitRoutingTbl(), and CFE_SB↩
_SendPrevSubsCmd().

12.143.2.13 SenderReporting

```
uint32 cfe_sb_t::SenderReporting
```

Definition at line 289 of file cfe_sb_priv.h.

Referenced by CFE_SB_EarlyInit(), and CFE_SB_SendMsgFull().

12.143.2.14 SharedDataMutexId

```
uint32 cfe_sb_t::SharedDataMutexId
```

Definition at line 287 of file cfe_sb_priv.h.

Referenced by CFE_SB_EarlyInit(), CFE_SB_LockSharedData(), and CFE_SB_UnlockSharedData().

12.143.2.15 StatTlmMsg

```
CFE_SB_StatsTlm_t cfe_sb_t::StatTlmMsg
```

Definition at line 295 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_EarlyInit(), CFE_S↩
B_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_PutDestinationBlk(), CFE_SB_RcvMsg(), CFE_SB↩
_ReturnBufferToPool(), CFE_SB_SendHKTlmCmd(), CFE_SB_SendMsgFull(), CFE_SB_SendStatsCmd(), CFE_S↩
B_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and
CFE_SB_ZeroCopyReleasePtr().

12.143.2.16 StopRecurseFlags

```
uint32 cfe_sb_t::StopRecurseFlags[CFE_PLATFORM_ES_MAX_APPLICATIONS]
```

Definition at line 291 of file cfe_sb_priv.h.

Referenced by CFE_SB_FinishSendEvent(), and CFE_SB_RequestToSendEvent().

12.143.2.17 SubRprtMsg

```
CFE_SB_SingleSubscriptionTlm_t cfe_sb_t::SubRprtMsg
```

Definition at line 302 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_SubscribeFull().

12.143.2.18 SubscriptionReporting

```
uint32 cfe_sb_t::SubscriptionReporting
```

Definition at line 288 of file cfe_sb_priv.h.

Referenced by CFE_SB_EarlyInit(), CFE_SB_SetSubscriptionReporting(), and CFE_SB_SubscribeFull().

12.143.2.19 ZeroCopyTail

```
void* cfe_sb_t::ZeroCopyTail
```

Definition at line 292 of file cfe_sb_priv.h.

Referenced by CFE_SB_EarlyInit(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseAppId(), and CFE_SB↩
_ZeroCopyReleaseDesc().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.144 CFE_SB_WriteFileInfoCmd_Payload_t Struct Reference

Write File Info Commands.

```
#include <cfe_sb_msg.h>
```

Data Fields

- char Filename [CFE_MISSION_MAX_PATH_LEN]

  Path and Filename of data to be loaded.

12.144.1 Detailed Description

This structure contains a generic definition used by three SB commands, 'Write Routing Info to File' CFE_SB_SEND↩_ROUTING_INFO_CC, 'Write Pipe Info to File' CFE_SB_SEND_PIPE_INFO_CC and 'Write Map Info to File' CFE_S↩B_SEND_MAP_INFO_CC.

Definition at line 492 of file cfe_sb_msg.h.

12.144.2 Field Documentation

12.144.2.1 Filename

```
char CFE_SB_WriteFileInfoCmd_Payload_t::Filename[CFE_MISSION_MAX_PATH_LEN]
```

Definition at line 493 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.145 CFE_SB_WriteFileInfoCmd_t Struct Reference

```
#include <cfe_sb_msg.h>
```

Data Fields

- CFE_SB_CmdHdr_t Hdr

  cFE Software Bus Command Message Header CFE_SB_CmdHdr_t
- CFE_SB_WriteFileInfoCmd_Payload_t Payload

12.145.1 Detailed Description

Definition at line 496 of file cfe_sb_msg.h.

12.145.2 Field Documentation

12.145.2.1 Hdr

`CFE_SB_CmdHdr_t CFE_SB_WriteFileInfoCmd_t::Hdr`

Definition at line 497 of file cfe_sb_msg.h.

12.145.2.2 Payload

`CFE_SB_WriteFileInfoCmd_Payload_t CFE_SB_WriteFileInfoCmd_t::Payload`

Definition at line 498 of file cfe_sb_msg.h.

Referenced by CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h

12.146 CFE_SB_ZeroCopyD_t Struct Reference

`#include <cfe_sb_priv.h>`

Data Fields

- uint32 AppID
- uint32 Size
- void * Buffer
- void * Next
- void * Prev

12.146.1 Detailed Description

Definition at line 217 of file cfe_sb_priv.h.

12.146.2 Field Documentation

12.146.2.1 AppID

```
uint32 CFE_SB_ZeroCopyD_t::AppID
```

Definition at line 218 of file cfe_sb_priv.h.

Referenced by CFE_SB_ZeroCopyReleaseAppId().

12.146.2.2 Buffer

```
void* CFE_SB_ZeroCopyD_t::Buffer
```

Definition at line 220 of file cfe_sb_priv.h.

Referenced by CFE_SB_ZeroCopyReleaseAppId().

12.146.2.3 Next

```
void* CFE_SB_ZeroCopyD_t::Next
```

Definition at line 221 of file cfe_sb_priv.h.

12.146.2.4 Prev

```
void* CFE_SB_ZeroCopyD_t::Prev
```

Definition at line 222 of file cfe_sb_priv.h.

Referenced by CFE_SB_ZeroCopyReleaseAppId().

12.146.2.5 Size

```
uint32 CFE_SB_ZeroCopyD_t::Size
```

Definition at line 219 of file cfe_sb_priv.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h

12.147 CFE_TBL_AbortLoad_t Struct Reference

`#include <cfe_tbl_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_AbortLoadCmd_Payload_t Payload

12.147.1 Detailed Description

Definition at line 666 of file cfe_tbl_msg.h.

12.147.2 Field Documentation

12.147.2.1 CmdHeader

`uint8 CFE_TBL_AbortLoad_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 668 of file cfe_tbl_msg.h.

12.147.2.2 Payload

`CFE_TBL_AbortLoadCmd_Payload_t CFE_TBL_AbortLoad_t::Payload`

Definition at line 669 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.148 CFE_TBL_AbortLoadCmd_Payload_t Struct Reference

Abort Load Command.

`#include <cfe_tbl_msg.h>`

Data Fields

- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
  Full Name of Table whose load is to be aborted.

12.148.1 Detailed Description

For command details, see CFE_TBL_ABORT_LOAD_CC

Definition at line 659 of file cfe_tbl_msg.h.

12.148.2 Field Documentation

12.148.2.1 TableName

```
char CFE_TBL_AbortLoadCmd_Payload_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

ASCII string containing full table name identifier of a table whose load is to be aborted

Definition at line 661 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.149 CFE_TBL_Activate_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_ActivateCmd_Payload_t Payload

12.149.1 Detailed Description

Definition at line 589 of file cfe_tbl_msg.h.

12.149.2 Field Documentation

12.149.2.1 CmdHeader

```
uint8 CFE_TBL_Activate_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 591 of file cfe_tbl_msg.h.

12.149.2.2 Payload

```
CFE_TBL_ActivateCmd_Payload_t CFE_TBL_Activate_t::Payload
```

Definition at line 592 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.150 CFE_TBL_ActivateCmd_Payload_t Struct Reference

Activate Table Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
  Full Name of Table to be activated.

12.150.1 Detailed Description

For command details, see CFE_TBL_ACTIVATE_CC

Definition at line 582 of file cfe_tbl_msg.h.

12.150.2 Field Documentation

12.150.2.1 TableName

```
char CFE_TBL_ActivateCmd_Payload_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

ASCII string containing full table name identifier of table to be activated

Definition at line 584 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.151 CFE_TBL_DelCDSCmd_Payload_t Struct Reference

Delete Critical Table CDS Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
  Full Name of Table whose CDS is to be deleted.

12.151.1 Detailed Description

For command details, see CFE_TBL_DELETE_CDS_CC

Definition at line 639 of file cfe_tbl_msg.h.

12.151.2 Field Documentation

12.151.2.1 TableName

```
char CFE_TBL_DelCDSCmd_Payload_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

ASCII string containing full table name identifier of a critical table whose CDS is to be deleted

Definition at line 641 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.152 CFE_TBL_DeleteCDS_t Struct Reference

`#include <cfe_tbl_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_DelCDSCmd_Payload_t Payload

12.152.1 Detailed Description

Definition at line 648 of file cfe_tbl_msg.h.

12.152.2 Field Documentation

12.152.2.1 CmdHeader

```
uint8 CFE_TBL_DeleteCDS_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 650 of file cfe_tbl_msg.h.

12.152.2.2 Payload

```
CFE_TBL_DelCDSCmd_Payload_t CFE_TBL_DeleteCDS_t::Payload
```

Definition at line 651 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.153 CFE_TBL_Dump_t Struct Reference

`#include <cfe_tbl_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_DumpCmd_Payload_t Payload

12.153.1 Detailed Description

Definition at line 547 of file cfe_tbl_msg.h.

12.153.2 Field Documentation

12.153.2.1 CmdHeader

```
uint8 CFE_TBL_Dump_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 549 of file cfe_tbl_msg.h.

12.153.2.2 Payload

```
CFE_TBL_DumpCmd_Payload_t CFE_TBL_Dump_t::Payload
```

Definition at line 550 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.154 CFE_TBL_DumpCmd_Payload_t Struct Reference

Dump Table Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint16 ActiveTableFlag

  CFE_TBL_BufferSelect_INACTIVE=Inactive Table, CFE_TBL_BufferSelect_ACTIVE=Active Table
- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Full name of table to be dumped.
- char DumpFilename [CFE_MISSION_MAX_PATH_LEN]

  Full Filename where data is to be written.

12.154.1 Detailed Description

For command details, see CFE_TBL_DUMP_CC

Definition at line 531 of file cfe_tbl_msg.h.

12.154.2 Field Documentation

12.154.2.1 ActiveTableFlag

```
uint16 CFE_TBL_DumpCmd_Payload_t::ActiveTableFlag
```

Selects either the "Inactive" (CFE_TBL_BufferSelect_INACTIVE) buffer or the "Active" (CFE_TBL_BufferSelect_ACTIVE) buffer to be dumped

Definition at line 533 of file cfe_tbl_msg.h.

12.154.2.2 DumpFilename

```
char CFE_TBL_DumpCmd_Payload_t::DumpFilename[CFE_MISSION_MAX_PATH_LEN]
```

ASCII string containing full path of filename where data is to be dumped

Definition at line 542 of file cfe_tbl_msg.h.

12.154.2.3 TableName

```
char CFE_TBL_DumpCmd_Payload_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

ASCII string containing full table name identifier of table to be dumped

Definition at line 539 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.155 CFE_TBL_DumpRegistry_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_DumpRegistryCmd_Payload_t Payload

12.155.1 Detailed Description

Definition at line 608 of file cfe_tbl_msg.h.

12.155.2 Field Documentation

12.155.2.1 CmdHeader

```
uint8 CFE_TBL_DumpRegistry_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 610 of file cfe_tbl_msg.h.

12.155.2.2 Payload

```
CFE_TBL_DumpRegistryCmd_Payload_t CFE_TBL_DumpRegistry_t::Payload
```

Definition at line 611 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.156 CFE_TBL_DumpRegistryCmd_Payload_t Struct Reference

Dump Registry Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char DumpFilename [CFE_MISSION_MAX_PATH_LEN]
  Full Filename where dumped data is to be written.

12.156.1 Detailed Description

For command details, see CFE_TBL_DUMP_REGISTRY_CC

Definition at line 600 of file cfe_tbl_msg.h.

12.156.2 Field Documentation

12.156.2.1 DumpFilename

```
char CFE_TBL_DumpRegistryCmd_Payload_t::DumpFilename[CFE_MISSION_MAX_PATH_LEN]
```

ASCII string containing full path of filename where registry is to be dumped

Definition at line 602 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.157 CFE_TBL_File_Hdr_t Struct Reference

The definition of the header fields that are included in CFE Table Data files.

```
#include <cfe_tbl_extern_typedefs.h>
```

Data Fields

- uint32 Reserved
- uint32 Offset
- uint32 NumBytes
- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

12.157.1 Detailed Description

This header follows the CFE_FS header and precedes the the actual table data.

Definition at line 69 of file cfe_tbl_extern_typedefs.h.

12.157.2 Field Documentation

12.157.2.1 NumBytes

```
uint32 CFE_TBL_File_Hdr_t::NumBytes
```

Number of bytes to load into table

Definition at line 73 of file cfe_tbl_extern_typedefs.h.

12.157.2.2 Offset

```
uint32 CFE_TBL_File_Hdr_t::Offset
```

Byte Offset at which load should commence

Definition at line 72 of file cfe_tbl_extern_typedefs.h.

12.157.2.3 Reserved

```
uint32 CFE_TBL_File_Hdr_t::Reserved
```

Future Use: NumTblSegments in File?

Definition at line 71 of file cfe_tbl_extern_typedefs.h.

12.157.2.4 TableName

```
char CFE_TBL_File_Hdr_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Fully qualified name of table to load

Definition at line 74 of file cfe_tbl_extern_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_extern_typedefs.h

12.158 CFE_TBL_FileDef_t Struct Reference

```
#include <cfe_tbl_filedef.h>
```

Data Fields

- char ObjectName [64]
  Name of instantiated variable that contains desired table image.
- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
  Name of Table as defined onboard.
- char Description [CFE_FS_HDR_DESC_MAX_LEN]
  Description of table image that is included in cFE File Header.
- char TgtFilename [CFE_MISSION_MAX_FILE_LEN]
  Default filename to be used for output of elf2cfetbl utility.
- uint32 ObjectSize
  Size, in bytes, of instantiated object.

12.158.1 Detailed Description

Definition at line 61 of file cfe_tbl_filedef.h.

12.158.2 Field Documentation

12.158.2.1 Description

```
char CFE_TBL_FileDef_t::Description[CFE_FS_HDR_DESC_MAX_LEN]
```

Definition at line 65 of file cfe_tbl_filedef.h.

12.158.2.2 ObjectName

```
char CFE_TBL_FileDef_t::ObjectName[64]
```

Definition at line 63 of file cfe_tbl_filedef.h.

12.158.2.3 ObjectSize

```
uint32 CFE_TBL_FileDef_t::ObjectSize
```

Definition at line 67 of file cfe_tbl_filedef.h.

12.158.2.4 TableName

```
char CFE_TBL_FileDef_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Definition at line 64 of file cfe_tbl_filedef.h.

12.158.2.5 TgtFilename

```
char CFE_TBL_FileDef_t::TgtFilename[CFE_MISSION_MAX_FILE_LEN]
```

Definition at line 66 of file cfe_tbl_filedef.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_filedef.h

12.159 CFE_TBL_HousekeepingTlm_Payload_t Struct Reference

`#include <cfe_tbl_msg.h>`

Data Fields

- uint8 CommandCounter

  Count of valid commands received.
- uint8 CommandErrorCounter

  Count of invalid commands received.
- uint16 NumTables

  Number of Tables Registered.
- uint16 NumLoadPending

  Number of Tables pending on Applications for their update.
- uint16 ValidationCounter

  Number of completed table validations.
- uint32 LastValCrc

  Data Integrity Value computed for last table validated.
- int32 LastValStatus

  Returned status from validation function for last table validated.
- bool ActiveBuffer

  Indicator of whether table buffer validated was 0=Inactive, 1=Active.
- char LastValTableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of last table validated.
- uint8 SuccessValCounter

  Total number of successful table validations.
- uint8 FailedValCounter

  Total number of unsuccessful table validations.
- uint8 NumValRequests

  Number of times Table Services has requested validations from Apps.
- uint8 NumFreeSharedBufs

  Number of free Shared Working Buffers.
- uint8 ByteAlignPad1

  Spare byte to ensure longword alignment.
- CFE_ES_MemHandle_t MemPoolHandle

  Handle to TBL's memory pool.
- CFE_TIME_SysTime_t LastUpdateTime

  Time of last table update.
- char LastUpdatedTable [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of the last table updated.
- char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]

  Path and Name of last table image file loaded.
- char LastFileDumped [CFE_MISSION_MAX_PATH_LEN]

  Path and Name of last file dumped to.
- char LastTableLoaded [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Name of the last table loaded.

12.159.1 Detailed Description

Name Table Services Housekeeping Packet

Definition at line 704 of file cfe_tbl_msg.h.

12.159.2 Field Documentation

12.159.2.1 ActiveBuffer

```
bool CFE_TBL_HousekeepingTlm_Payload_t::ActiveBuffer
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastValBuf**

Definition at line 731 of file cfe_tbl_msg.h.

12.159.2.2 ByteAlignPad1

```
uint8 CFE_TBL_HousekeepingTlm_Payload_t::ByteAlignPad1
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_ByteAlignPad1**

Definition at line 747 of file cfe_tbl_msg.h.

12.159.2.3 CommandCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload_t::CommandCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_CMDPC**

Definition at line 709 of file cfe_tbl_msg.h.

12.159.2.4 CommandErrorCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload_t::CommandErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_CMDEC**

Definition at line 711 of file cfe_tbl_msg.h.

12.159.2.5 FailedValCounter

```
uint8 CFE_TBL_HousekeepingTlm_Payload_t::FailedValCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_ValFailedCtr**

Definition at line 737 of file cfe_tbl_msg.h.

12.159.2.6 LastFileDumped

```
char CFE_TBL_HousekeepingTlm_Payload_t::LastFileDumped[CFE_MISSION_MAX_PATH_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastFileDumped[OS_MAX_PATH_LEN]**

Definition at line 757 of file cfe_tbl_msg.h.

12.159.2.7 LastFileLoaded

```
char CFE_TBL_HousekeepingTlm_Payload_t::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastFileLoaded[OS_MAX_PATH_LEN]**

Definition at line 755 of file cfe_tbl_msg.h.

12.159.2.8 LastTableLoaded

```
char CFE_TBL_HousekeepingTlm_Payload_t::LastTableLoaded[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastTableLoaded[CFE_TBL_MAX_FULL_NAME_LEN]**

Definition at line 759 of file cfe_tbl_msg.h.

12.159.2.9 LastUpdatedTable

```
char CFE_TBL_HousekeepingTlm_Payload_t::LastUpdatedTable[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastUpdTblName[CFE_TB_MAX_FULL_NAME_LEN]**

Definition at line 753 of file cfe_tbl_msg.h.

12.159.2.10 LastUpdateTime

```
CFE_TIME_SysTime_t CFE_TBL_HousekeepingTlm_Payload_t::LastUpdateTime
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastUpdTime, $sc_$cpu_TBL_SECONDS, $sc_$cpu_TBL_S↩ UBSECONDS**

Definition at line 751 of file cfe_tbl_msg.h.

12.159.2.11 LastValCrc

```
uint32 CFE_TBL_HousekeepingTlm_Payload_t::LastValCrc
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastValCRC**

Definition at line 727 of file cfe_tbl_msg.h.

12.159.2.12 LastValStatus

```
int32 CFE_TBL_HousekeepingTlm_Payload_t::LastValStatus
```

Telemetry Mnemonic(s) **$sc_$cpu_TBl_LastValS**

Definition at line 729 of file cfe_tbl_msg.h.

12.159.2.13 LastValTableName

```
char CFE_TBL_HousekeepingTlm_Payload_t::LastValTableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastValTblName[CFE_TB_MAX_FULL_NAME_LEN]**

Definition at line 733 of file cfe_tbl_msg.h.

12.159.2.14 MemPoolHandle

```
CFE_ES_MemHandle_t CFE_TBL_HousekeepingTlm_Payload_t::MemPoolHandle
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_MemPoolHandle**

Definition at line 749 of file cfe_tbl_msg.h.

12.159.2.15 NumFreeSharedBufs

```
uint8 CFE_TBL_HousekeepingTlm_Payload_t::NumFreeSharedBufs
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_NumFreeShrBuf**

Definition at line 745 of file cfe_tbl_msg.h.

12.159.2.16 NumLoadPending

uint16 CFE_TBL_HousekeepingTlm_Payload_t::NumLoadPending

Telemetry Mnemonic(s) **$sc_$cpu_TBL_NumUpdatesPend**

Definition at line 719 of file cfe_tbl_msg.h.

12.159.2.17 NumTables

uint16 CFE_TBL_HousekeepingTlm_Payload_t::NumTables

Telemetry Mnemonic(s) **$sc_$cpu_TBL_NumTables**

Definition at line 717 of file cfe_tbl_msg.h.

12.159.2.18 NumValRequests

uint8 CFE_TBL_HousekeepingTlm_Payload_t::NumValRequests

Telemetry Mnemonic(s) **$sc_$cpu_TBL_ValReqCtr**

Definition at line 739 of file cfe_tbl_msg.h.

12.159.2.19 SuccessValCounter

uint8 CFE_TBL_HousekeepingTlm_Payload_t::SuccessValCounter

Telemetry Mnemonic(s) **$sc_$cpu_TBL_ValSuccessCtr**

Definition at line 735 of file cfe_tbl_msg.h.

12.159.2.20 ValidationCounter

```
uint16 CFE_TBL_HousekeepingTlm_Payload_t::ValidationCounter
```

Telemetry Mnemonic(s) **`$sc_$cpu_TBL_ValCompltdCtr`**

Definition at line 725 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.160 CFE_TBL_HousekeepingTlm_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
  cFE Software Bus Telemetry Message Header
- CFE_TBL_HousekeepingTlm_Payload_t Payload

12.160.1 Detailed Description

Definition at line 763 of file cfe_tbl_msg.h.

12.160.2 Field Documentation

12.160.2.1 Payload

```
CFE_TBL_HousekeepingTlm_Payload_t CFE_TBL_HousekeepingTlm_t::Payload
```

Definition at line 766 of file cfe_tbl_msg.h.

12.160.2.2 TlmHeader

```
uint8 CFE_TBL_HousekeepingTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 765 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.161 CFE_TBL_Info_t Struct Reference

`#include <cfe_tbl.h>`

Data Fields

- uint32 Size

  Size, in bytes, of Table.
- uint32 NumUsers

  Number of Apps with access to the table.
- uint32 FileCreateTimeSecs

  File creation time from last file loaded into table.
- uint32 FileCreateTimeSubSecs

  File creation time from last file loaded into table.
- uint32 Crc

  Most recently calculated CRC by TBL services on table contents.
- CFE_TIME_SysTime_t TimeOfLastUpdate

  Time when Table was last updated.
- bool TableLoadedOnce

  Flag indicating whether table has been loaded once or not.
- bool DumpOnly

  Flag indicating Table is NOT to be loaded.
- bool DoubleBuffered

  Flag indicating Table has a dedicated inactive buffer.
- bool UserDefAddr

  Flag indicating Table address was defined by Owner Application.
- bool Critical

  Flag indicating Table contents are maintained in a CDS.
- char LastFileLoaded [OS_MAX_PATH_LEN]

  Filename of last file loaded into table.

12.161.1 Detailed Description

Definition at line 117 of file cfe_tbl.h.

12.161.2 Field Documentation

12.161.2.1 Crc

`uint32 CFE_TBL_Info_t::Crc`

Definition at line 123 of file cfe_tbl.h.

12.161.2.2 Critical

```
bool CFE_TBL_Info_t::Critical
```

Definition at line 129 of file cfe_tbl.h.

12.161.2.3 DoubleBuffered

```
bool CFE_TBL_Info_t::DoubleBuffered
```

Definition at line 127 of file cfe_tbl.h.

12.161.2.4 DumpOnly

```
bool CFE_TBL_Info_t::DumpOnly
```

Definition at line 126 of file cfe_tbl.h.

12.161.2.5 FileCreateTimeSecs

```
uint32 CFE_TBL_Info_t::FileCreateTimeSecs
```

Definition at line 121 of file cfe_tbl.h.

12.161.2.6 FileCreateTimeSubSecs

```
uint32 CFE_TBL_Info_t::FileCreateTimeSubSecs
```

Definition at line 122 of file cfe_tbl.h.

12.161.2.7 LastFileLoaded

```
char CFE_TBL_Info_t::LastFileLoaded[OS_MAX_PATH_LEN]
```

Definition at line 130 of file cfe_tbl.h.

12.161.2.8 NumUsers

```
uint32 CFE_TBL_Info_t::NumUsers
```

Definition at line 120 of file cfe_tbl.h.

12.161.2.9 Size

```
uint32 CFE_TBL_Info_t::Size
```

Definition at line 119 of file cfe_tbl.h.

12.161.2.10 TableLoadedOnce

```
bool CFE_TBL_Info_t::TableLoadedOnce
```

Definition at line 125 of file cfe_tbl.h.

12.161.2.11 TimeOfLastUpdate

```
CFE_TIME_SysTime_t CFE_TBL_Info_t::TimeOfLastUpdate
```

Definition at line 124 of file cfe_tbl.h.

12.161.2.12 UserDefAddr

```
bool CFE_TBL_Info_t::UserDefAddr
```

Definition at line 128 of file cfe_tbl.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl.h

12.162 CFE_TBL_Load_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_LoadCmd_Payload_t Payload

12.162.1 Detailed Description

Definition at line 520 of file cfe_tbl_msg.h.

12.162.2 Field Documentation

12.162.2.1 CmdHeader

```
uint8 CFE_TBL_Load_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 522 of file cfe_tbl_msg.h.

12.162.2.2 Payload

```
CFE_TBL_LoadCmd_Payload_t CFE_TBL_Load_t::Payload
```

Definition at line 523 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.163 CFE_TBL_LoadCmd_Payload_t Struct Reference

Load Table Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char LoadFilename [CFE_MISSION_MAX_PATH_LEN]
  Filename (and path) of data to be loaded.

12.163.1 Detailed Description

For command details, see CFE_TBL_LOAD_CC

Definition at line 513 of file cfe_tbl_msg.h.

12.163.2 Field Documentation

12.163.2.1 LoadFilename

```
char CFE_TBL_LoadCmd_Payload_t::LoadFilename[CFE_MISSION_MAX_PATH_LEN]
```

ASCII Character string containing full path filename for file to be loaded

Definition at line 515 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.164 CFE_TBL_NoArgsCmd_t Struct Reference

Generic "no arguments" command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header

12.164.1 Detailed Description

This command structure is used for commands that do not have any parameters. This includes:

1. The Housekeeping Request Message
2. The No-Op Command (For details, see CFE_TBL_NOOP_CC)
3. The Reset Counters Command (For details, see CFE_TBL_RESET_COUNTERS_CC)

Definition at line 493 of file cfe_tbl_msg.h.

12.164.2 Field Documentation

12.164.2.1 CmdHeader

```
uint8 CFE_TBL_NoArgsCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 495 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.165 CFE_TBL_NotifyCmd_Payload_t Struct Reference

Table Management Notification Message.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint32 Parameter

  Application specified command parameter.

12.165.1 Detailed Description

Description

Whenever an application that owns a table calls the CFE_TBL_NotifyByMessage API following the table registration, Table services will generate the following command message with the application specified message ID, command code and parameter whenever the table requires management (e.g. - loads and validations).

Definition at line 686 of file cfe_tbl_msg.h.

12.165.2 Field Documentation

12.165.2.1 Parameter

```
uint32 CFE_TBL_NotifyCmd_Payload_t::Parameter
```

Definition at line 688 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.166 CFE_TBL_NotifyCmd_t Struct Reference

`#include <cfe_tbl_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_NotifyCmd_Payload_t Payload

12.166.1 Detailed Description

Definition at line 691 of file cfe_tbl_msg.h.

12.166.2 Field Documentation

12.166.2.1 CmdHeader

```
uint8 CFE_TBL_NotifyCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 693 of file cfe_tbl_msg.h.

12.166.2.2 Payload

```
CFE_TBL_NotifyCmd_Payload_t CFE_TBL_NotifyCmd_t::Payload
```

Definition at line 694 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.167 CFE_TBL_SendRegistry_t Struct Reference

`#include <cfe_tbl_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_SendRegistryCmd_Payload_t Payload

12.167.1 Detailed Description

Definition at line 628 of file cfe_tbl_msg.h.

12.167.2 Field Documentation

12.167.2.1 CmdHeader

```
uint8 CFE_TBL_SendRegistry_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 630 of file cfe_tbl_msg.h.

12.167.2.2 Payload

```
CFE_TBL_SendRegistryCmd_Payload_t CFE_TBL_SendRegistry_t::Payload
```

Definition at line 631 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.168 CFE_TBL_SendRegistryCmd_Payload_t Struct Reference

Telemeter Table Registry Entry Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Full Name of Table whose registry entry is to be telemetered.

12.168.1 Detailed Description

For command details, see CFE_TBL_SEND_REGISTRY_CC

Definition at line 619 of file cfe_tbl_msg.h.

12.168.2 Field Documentation

12.168.2.1 TableName

```
char CFE_TBL_SendRegistryCmd_Payload_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

ASCII string containing full table name identifier of table whose registry entry is to be telemetered via CFE_TBL_Table↩ RegistryTlm_t

Definition at line 621 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.169 CFE_TBL_TableRegistryTlm_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
  cFE Software Bus Telemetry Message Header
- CFE_TBL_TblRegPacket_Payload_t Payload

12.169.1 Detailed Description

Definition at line 811 of file cfe_tbl_msg.h.

12.169.2 Field Documentation

12.169.2.1 Payload

```
CFE_TBL_TblRegPacket_Payload_t CFE_TBL_TableRegistryTlm_t::Payload
```

Definition at line 814 of file cfe_tbl_msg.h.

12.169.2.2 TlmHeader

```
uint8 CFE_TBL_TableRegistryTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 813 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.170 CFE_TBL_TblRegPacket_Payload_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint32 Size
  Size, in bytes, of Table.
- uint32 Crc
  Most recently calculated CRC of Table.
- cpuaddr ActiveBufferAddr
  Address of Active Buffer.
- cpuaddr InactiveBufferAddr
  Address of Inactive Buffer.
- cpuaddr ValidationFuncPtr
  Ptr to Owner App's function that validates tbl contents.
- CFE_TIME_SysTime_t TimeOfLastUpdate
  Time when Table was last updated.
- uint32 FileCreateTimeSecs
  File creation time from last file loaded into table.
- uint32 FileCreateTimeSubSecs
  File creation time from last file loaded into table.
- bool TableLoadedOnce
  Flag indicating whether table has been loaded once or not.
- bool LoadPending
  Flag indicating an inactive buffer is ready to be copied.
- bool DumpOnly
  Flag indicating Table is NOT to be loaded.
- bool DoubleBuffered
  Flag indicating Table has a dedicated inactive buffer.
- char Name [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
  Processor specific table name.
- char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]
  Filename of last file loaded into table.
- char OwnerAppName [CFE_MISSION_MAX_API_LEN]
  Name of owning application.
- bool Critical
  Indicates whether table is Critical or not.
- uint8 ByteAlign4
  Spare byte to maintain byte alignment.

12.170.1 Detailed Description

Name Table Registry Info Packet

Definition at line 773 of file cfe_tbl_msg.h.

12.170.2 Field Documentation

12.170.2.1 ActiveBufferAddr

```
cpuaddr CFE_TBL_TblRegPacket_Payload_t::ActiveBufferAddr
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_ActBufAdd**

Definition at line 779 of file cfe_tbl_msg.h.

12.170.2.2 ByteAlign4

```
uint8 CFE_TBL_TblRegPacket_Payload_t::ByteAlign4
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_Spare4**

Definition at line 807 of file cfe_tbl_msg.h.

12.170.2.3 Crc

```
uint32 CFE_TBL_TblRegPacket_Payload_t::Crc
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_CRC**

Definition at line 777 of file cfe_tbl_msg.h.

12.170.2.4 Critical

```
bool CFE_TBL_TblRegPacket_Payload_t::Critical
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_Spare3**

Definition at line 805 of file cfe_tbl_msg.h.

12.170.2.5 DoubleBuffered

```
bool CFE_TBL_TblRegPacket_Payload_t::DoubleBuffered
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_DblBuffered**

Definition at line 797 of file cfe_tbl_msg.h.

12.170.2.6 DumpOnly

```
bool CFE_TBL_TblRegPacket_Payload_t::DumpOnly
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_DumpOnly**

Definition at line 795 of file cfe_tbl_msg.h.

12.170.2.7 FileCreateTimeSecs

```
uint32 CFE_TBL_TblRegPacket_Payload_t::FileCreateTimeSecs
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_FILECSECONDS**

Definition at line 787 of file cfe_tbl_msg.h.

12.170.2.8 FileCreateTimeSubSecs

```
uint32 CFE_TBL_TblRegPacket_Payload_t::FileCreateTimeSubSecs
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_FILECSUBSECONDS**

Definition at line 789 of file cfe_tbl_msg.h.

12.170.2.9 InactiveBufferAddr

```
cpuaddr CFE_TBL_TblRegPacket_Payload_t::InactiveBufferAddr
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_IActBufAdd**

Definition at line 781 of file cfe_tbl_msg.h.

12.170.2.10 LastFileLoaded

```
char CFE_TBL_TblRegPacket_Payload_t::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LastFileUpd[OS_MAX_PATH_LEN]**

Definition at line 801 of file cfe_tbl_msg.h.

12.170.2.11 LoadPending

```
bool CFE_TBL_TblRegPacket_Payload_t::LoadPending
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_UpdatePndng**

Definition at line 793 of file cfe_tbl_msg.h.

12.170.2.12 Name

```
char CFE_TBL_TblRegPacket_Payload_t::Name[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_Name[CFE_TB_MAX_FULL_NAME_LEN]**

Definition at line 799 of file cfe_tbl_msg.h.

12.170.2.13 OwnerAppName

```
char CFE_TBL_TblRegPacket_Payload_t::OwnerAppName[CFE_MISSION_MAX_API_LEN]
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_OwnerApp[OS_MAX_API_NAME]**

Definition at line 803 of file cfe_tbl_msg.h.

12.170.2.14 Size

```
uint32 CFE_TBL_TblRegPacket_Payload_t::Size
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_SIZE**

Definition at line 775 of file cfe_tbl_msg.h.

12.170.2.15 TableLoadedOnce

```
bool CFE_TBL_TblRegPacket_Payload_t::TableLoadedOnce
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_LoadedOnce**

Definition at line 791 of file cfe_tbl_msg.h.

12.170.2.16 TimeOfLastUpdate

```
CFE_TIME_SysTime_t CFE_TBL_TblRegPacket_Payload_t::TimeOfLastUpdate
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_TimeLastUpd, $sc_$cpu_TBL_TLUSECONDS, $sc_$cpu_TBL_TLUSUBSECONDS**

Definition at line 785 of file cfe_tbl_msg.h.

12.170.2.17 ValidationFuncPtr

```
cpuaddr CFE_TBL_TblRegPacket_Payload_t::ValidationFuncPtr
```

Telemetry Mnemonic(s) **$sc_$cpu_TBL_ValFuncPtr**

Definition at line 783 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.171 CFE_TBL_Validate_t Struct Reference

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
  cFE Software Bus Command Message Header
- CFE_TBL_ValidateCmd_Payload_t Payload

12.171.1 Detailed Description

Definition at line 571 of file cfe_tbl_msg.h.

12.171.2 Field Documentation

12.171.2.1 CmdHeader

```
uint8 CFE_TBL_Validate_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 573 of file cfe_tbl_msg.h.

12.171.2.2 Payload

```
CFE_TBL_ValidateCmd_Payload_t CFE_TBL_Validate_t::Payload
```

Definition at line 574 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.172 CFE_TBL_ValidateCmd_Payload_t Struct Reference

Validate Table Command.

```
#include <cfe_tbl_msg.h>
```

Data Fields

- uint16 ActiveTableFlag

  CFE_TBL_BufferSelect_INACTIVE=Inactive Table, CFE_TBL_BufferSelect_ACTIVE=Active Table
- char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]

  Full Name of Table to be validated.

12.172.1 Detailed Description

For command details, see CFE_TBL_VALIDATE_CC

Definition at line 558 of file cfe_tbl_msg.h.

12.172.2 Field Documentation

12.172.2.1 ActiveTableFlag

```
uint16 CFE_TBL_ValidateCmd_Payload_t::ActiveTableFlag
```

Selects either the "Inactive" (CFE_TBL_BufferSelect_INACTIVE) buffer or the "Active" (CFE_TBL_BufferSelect_ACTIVE) buffer to be validated

Definition at line 560 of file cfe_tbl_msg.h.

12.172.2.2 TableName

```
char CFE_TBL_ValidateCmd_Payload_t::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]
```

ASCII string containing full table name identifier of table to be validated

Definition at line 566 of file cfe_tbl_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h

12.173 CFE_TIME_1HzCmd_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

12.173.1 Detailed Description

Definition at line 871 of file cfe_time_msg.h.

12.173.2 Field Documentation

12.173.2.1 CmdHeader

```
uint8 CFE_TIME_1HzCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 873 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.174 CFE_TIME_DiagnosticTlm_Payload_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- CFE_TIME_SysTime_t AtToneMET
  MET at time of tone.
- CFE_TIME_SysTime_t AtToneSTCF
  STCF at time of tone.
- CFE_TIME_SysTime_t AtToneDelay
  Adjustment for slow tone detection.
- CFE_TIME_SysTime_t AtToneLatch
  Local clock latched at time of tone.
- int16 AtToneLeapSeconds
  Leap Seconds at time of tone.
- int16 ClockStateAPI
  Clock state as per API.
- CFE_TIME_SysTime_t TimeSinceTone
  Time elapsed since the tone.
- CFE_TIME_SysTime_t CurrentLatch
  Local clock latched just "now".
- CFE_TIME_SysTime_t CurrentMET
  MET at this instant.
- CFE_TIME_SysTime_t CurrentTAI
  TAI at this instant.
- CFE_TIME_SysTime_t CurrentUTC
  UTC at this instant.
- int16 ClockSetState
  Time has been "set".
- int16 ClockFlyState
  Current fly-wheel state.
- int16 ClockSource
  Internal vs external, etc.
- int16 ClockSignal
  Primary vs redundant, etc.
- int16 ServerFlyState
  Used by clients only.
- int16 Forced2Fly
  Commanded into fly-wheel.
- uint16 ClockStateFlags
  Clock State Flags.
- int16 OneTimeDirection
  One time STCF adjustment direction (Add = 1, Sub = 2)
- int16 OneHzDirection
  1Hz STCF adjustment direction

- int16 DelayDirection

  Client latency adjustment direction.
- CFE_TIME_SysTime_t OneTimeAdjust

  Previous one-time STCF adjustment.
- CFE_TIME_SysTime_t OneHzAdjust

  Current 1Hz STCF adjustment.
- CFE_TIME_SysTime_t ToneSignalLatch

  Local Clock latched at most recent tone signal.
- CFE_TIME_SysTime_t ToneDataLatch

  Local Clock latched at arrival of tone data.
- uint32 ToneMatchCounter

  Tone signal / data verification count.
- uint32 ToneMatchErrorCounter

  Tone signal / data verification error count.
- uint32 ToneSignalCounter

  Tone signal detected SB message count.
- uint32 ToneDataCounter

  Time at the tone data SB message count.
- uint32 ToneIntCounter

  Tone signal ISR execution count.
- uint32 ToneIntErrorCounter

  Tone signal ISR error count.
- uint32 ToneTaskCounter

  Tone task execution count.
- uint32 VersionCounter

  Count of mods to time at tone reference data (version)
- uint32 LocalIntCounter

  Local 1Hz ISR execution count.
- uint32 LocalTaskCounter

  Local 1Hz task execution count.
- uint32 VirtualMET

  Software MET.
- uint32 MinElapsed

  Min tone signal / data pkt arrival window (Sub-seconds)
- uint32 MaxElapsed

  Max tone signal / data pkt arrival window (Sub-seconds)
- CFE_TIME_SysTime_t MaxLocalClock

  Max local clock value before rollover.
- uint32 ToneOverLimit

  Max between tone signal interrupts.
- uint32 ToneUnderLimit

  Min between tone signal interrupts.
- uint32 DataStoreStatus

  Data Store status (preserved across processor reset)

12.174.1 Detailed Description

Name Time Services Diagnostics Packet

Definition at line 992 of file cfe_time_msg.h.

12.174.2 Field Documentation

12.174.2.1 AtToneDelay

```
CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::AtToneDelay
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DLatentS, $sc_$cpu_TIME_DLatentSs**

Definition at line 1001 of file cfe_time_msg.h.

12.174.2.2 AtToneLatch

```
CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::AtToneLatch
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTValidS, $sc_$cpu_TIME_DTValidSs**

Definition at line 1003 of file cfe_time_msg.h.

12.174.2.3 AtToneLeapSeconds

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::AtToneLeapSeconds
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DLeapS**

Definition at line 1006 of file cfe_time_msg.h.

12.174.2.4 AtToneMET

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::AtToneMET

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTMETS, $sc_$cpu_TIME_DTMETSs**

Definition at line 997 of file cfe_time_msg.h.

12.174.2.5 AtToneSTCF

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::AtToneSTCF

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DSTCFS, $sc_$cpu_TIME_DSTCFSS**

Definition at line 999 of file cfe_time_msg.h.

12.174.2.6 ClockFlyState

int16 CFE_TIME_DiagnosticTlm_Payload_t::ClockFlyState

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DFlywheel**

Definition at line 1030 of file cfe_time_msg.h.

12.174.2.7 ClockSetState

int16 CFE_TIME_DiagnosticTlm_Payload_t::ClockSetState

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DValid**

Definition at line 1028 of file cfe_time_msg.h.

12.174.2.8 ClockSignal

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::ClockSignal
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DSignal**

Definition at line 1034 of file cfe_time_msg.h.

12.174.2.9 ClockSource

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::ClockSource
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DSource**

Definition at line 1032 of file cfe_time_msg.h.

12.174.2.10 ClockStateAPI

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::ClockStateAPI
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DAPIState**

Definition at line 1008 of file cfe_time_msg.h.

12.174.2.11 ClockStateFlags

```
uint16 CFE_TIME_DiagnosticTlm_Payload_t::ClockStateFlags
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DStateFlags, $sc_$cpu_TIME_DFlagSet, $sc_$cpu_TIME_DFlagFly, $sc_$cpu_TIME_DFlagSrc, $sc_$cpu_TIME_DFlagPri, $sc_$cpu_TIME_DFlagSfly, $sc_$cpu_TIME_DFlagCfly, $sc_$cpu_TIME_DFlagAdjd, $sc_$cpu_TIME_DFlag1Hzd, $sc_$cpu_TIME_DFlagClat, $sc_$cpu_TIME_DFlagSorC, $sc_$cpu_TIME_DFlagNIU**

Definition at line 1044 of file cfe_time_msg.h.

12.174.2.12 CurrentLatch

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::CurrentLatch

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DLocalS, $sc_$cpu_TIME_DLocalSs**

Definition at line 1016 of file cfe_time_msg.h.

12.174.2.13 CurrentMET

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::CurrentMET

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DMETS, $sc_$cpu_TIME_DMETSs**

Definition at line 1018 of file cfe_time_msg.h.

12.174.2.14 CurrentTAI

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::CurrentTAI

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTAIS, $sc_$cpu_TIME_DTAISS**

Definition at line 1020 of file cfe_time_msg.h.

12.174.2.15 CurrentUTC

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::CurrentUTC

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DUTCS, $sc_$cpu_TIME_DUTCSS**

Definition at line 1022 of file cfe_time_msg.h.

12.174.2.16 DataStoreStatus

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::DataStoreStatus
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DataStStat**

Definition at line 1134 of file cfe_time_msg.h.

12.174.2.17 DelayDirection

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::DelayDirection
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DLatentDir**

Definition at line 1054 of file cfe_time_msg.h.

12.174.2.18 Forced2Fly

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::Forced2Fly
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DCMD2Fly**

Definition at line 1038 of file cfe_time_msg.h.

12.174.2.19 LocalIntCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::LocalIntCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_D1HzISRCNT**

Definition at line 1092 of file cfe_time_msg.h.

12.174.2.20 LocalTaskCounter

uint32 CFE_TIME_DiagnosticTlm_Payload_t::LocalTaskCounter

Telemetry Mnemonic(s) **$sc_$cpu_TIME_D1HzTaskCNT**

Definition at line 1094 of file cfe_time_msg.h.

12.174.2.21 MaxElapsed

uint32 CFE_TIME_DiagnosticTlm_Payload_t::MaxElapsed

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DMaxWindow**

Definition at line 1114 of file cfe_time_msg.h.

12.174.2.22 MaxLocalClock

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::MaxLocalClock

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DWrapS**, **$sc_$cpu_TIME_DWrapSs**

Definition at line 1120 of file cfe_time_msg.h.

12.174.2.23 MinElapsed

uint32 CFE_TIME_DiagnosticTlm_Payload_t::MinElapsed

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DMinWindow**

Definition at line 1112 of file cfe_time_msg.h.

12.174.2.24 OneHzAdjust

```
CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::OneHzAdjust
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_D1HzAdjS, $sc_$cpu_TIME_D1HzAdjSs**

Definition at line 1062 of file cfe_time_msg.h.

12.174.2.25 OneHzDirection

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::OneHzDirection
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_D1HzAdjDir**

Definition at line 1052 of file cfe_time_msg.h.

12.174.2.26 OneTimeAdjust

```
CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::OneTimeAdjust
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DAdjustS, $sc_$cpu_TIME_DAdjustSs**

Definition at line 1060 of file cfe_time_msg.h.

12.174.2.27 OneTimeDirection

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::OneTimeDirection
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DAdjustDir**

Definition at line 1050 of file cfe_time_msg.h.

12.174.2.28 ServerFlyState

```
int16 CFE_TIME_DiagnosticTlm_Payload_t::ServerFlyState
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DSrvFly**

Definition at line 1036 of file cfe_time_msg.h.

12.174.2.29 TimeSinceTone

```
CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::TimeSinceTone
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DElapsedS**, **$sc_$cpu_TIME_DElapsedSs**

Definition at line 1014 of file cfe_time_msg.h.

12.174.2.30 ToneDataCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneDataCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTatTCNT**

Definition at line 1082 of file cfe_time_msg.h.

12.174.2.31 ToneDataLatch

```
CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::ToneDataLatch
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTDS**, **$sc_$cpu_TIME_DTDSs**

Definition at line 1070 of file cfe_time_msg.h.

12.174.2.32 ToneIntCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneIntCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTsISRCNT**

Definition at line 1084 of file cfe_time_msg.h.

12.174.2.33 ToneIntErrorCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneIntErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTsISRERR**

Definition at line 1086 of file cfe_time_msg.h.

12.174.2.34 ToneMatchCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneMatchCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DVerifyCNT**

Definition at line 1076 of file cfe_time_msg.h.

12.174.2.35 ToneMatchErrorCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneMatchErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DVerifyER**

Definition at line 1078 of file cfe_time_msg.h.

12.174.2.36 ToneOverLimit

uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneOverLimit

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DMaxSs**

Definition at line 1126 of file cfe_time_msg.h.

12.174.2.37 ToneSignalCounter

uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneSignalCounter

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTSDetCNT**

Definition at line 1080 of file cfe_time_msg.h.

12.174.2.38 ToneSignalLatch

CFE_TIME_SysTime_t CFE_TIME_DiagnosticTlm_Payload_t::ToneSignalLatch

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTTS**, **$sc_$cpu_TIME_DTTSs**

Definition at line 1068 of file cfe_time_msg.h.

12.174.2.39 ToneTaskCounter

uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneTaskCounter

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DTsTaskCNT**

Definition at line 1088 of file cfe_time_msg.h.

12.174.2.40 ToneUnderLimit

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::ToneUnderLimit
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DMinSs**

Definition at line 1128 of file cfe_time_msg.h.

12.174.2.41 VersionCounter

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::VersionCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DVersionCNT**

Definition at line 1090 of file cfe_time_msg.h.

12.174.2.42 VirtualMET

```
uint32 CFE_TIME_DiagnosticTlm_Payload_t::VirtualMET
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DLogicalMET**

Definition at line 1100 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.175 CFE_TIME_DiagnosticTlm_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
- CFE_TIME_DiagnosticTlm_Payload_t Payload

12.175.1 Detailed Description

Definition at line 1138 of file cfe_time_msg.h.

12.175.2 Field Documentation

12.175.2.1 Payload

```
CFE_TIME_DiagnosticTlm_Payload_t CFE_TIME_DiagnosticTlm_t::Payload
```

Definition at line 1141 of file cfe_time_msg.h.

12.175.2.2 TlmHeader

```
uint8 CFE_TIME_DiagnosticTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 1140 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.176 CFE_TIME_FakeToneCmd_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

12.176.1 Detailed Description

Definition at line 891 of file cfe_time_msg.h.

12.176.2 Field Documentation

12.176.2.1 CmdHeader

```
uint8 CFE_TIME_FakeToneCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 893 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.177 CFE_TIME_HousekeepingTlm_Payload_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CommandCounter

  Time Command Execution Counter.
- uint8 CommandErrorCounter

  Time Command Error Counter.
- uint16 ClockStateFlags

  State Flags.
- int16 ClockStateAPI

  API State.
- int16 LeapSeconds

  Current Leaps Seconds.
- uint32 SecondsMET

  Current MET (seconds)
- uint32 SubsecsMET

  Current MET (sub-seconds)
- uint32 SecondsSTCF

  Current STCF (seconds)
- uint32 SubsecsSTCF

  Current STCF (sub-seconds)
- uint32 Seconds1HzAdj

  Current 1 Hz SCTF adjustment (seconds)
- uint32 Subsecs1HzAdj

  Current 1 Hz SCTF adjustment (sub-seconds)
- uint32 SecondsDelay

  Current 1 Hz SCTF Delay (seconds)
- uint32 SubsecsDelay

  Current 1 Hz SCTF Delay (sub-seconds)

12.177.1 Detailed Description

Name Time Services Housekeeping Packet

Definition at line 921 of file cfe_time_msg.h.

12.177.2 Field Documentation

12.177.2.1 ClockStateAPI

```
int16 CFE_TIME_HousekeepingTlm_Payload_t::ClockStateAPI
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_DAPIState**

Definition at line 936 of file cfe_time_msg.h.

12.177.2.2 ClockStateFlags

```
uint16 CFE_TIME_HousekeepingTlm_Payload_t::ClockStateFlags
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_StateFlg, $sc_$cpu_TIME_FlagSet, $sc_$cpu_TIME_↩ FlagFly, $sc_$cpu_TIME_FlagSrc, $sc_$cpu_TIME_FlagPri, $sc_$cpu↩ _TIME_FlagSfly, $sc_$cpu_TIME_FlagCfly, $sc_$cpu_TIME_FlagAdjd, $sc_$cpu_TIME_Flag1Hzd, $sc_$cpu_TIME_FlagClat, $sc_$cpu_TIME_↩ FlagSorC, $sc_$cpu_TIME_FlagNIU**

Definition at line 934 of file cfe_time_msg.h.

12.177.2.3 CommandCounter

```
uint8 CFE_TIME_HousekeepingTlm_Payload_t::CommandCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_CMDPC**

Definition at line 926 of file cfe_time_msg.h.

12.177.2.4 CommandErrorCounter

```
uint8 CFE_TIME_HousekeepingTlm_Payload_t::CommandErrorCounter
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_CMDEC**

Definition at line 928 of file cfe_time_msg.h.

12.177.2.5 LeapSeconds

```
int16 CFE_TIME_HousekeepingTlm_Payload_t::LeapSeconds
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_LeapSecs**

Definition at line 942 of file cfe_time_msg.h.

12.177.2.6 Seconds1HzAdj

```
uint32 CFE_TIME_HousekeepingTlm_Payload_t::Seconds1HzAdj
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_1HzAdjSecs**

Definition at line 962 of file cfe_time_msg.h.

12.177.2.7 SecondsDelay

```
uint32 CFE_TIME_HousekeepingTlm_Payload_t::SecondsDelay
```

Telemetry Mnemonic(s) **$sc_$cpu_TIME_1HzAdjSecs**

Definition at line 972 of file cfe_time_msg.h.

12.177.2.8 SecondsMET

uint32 CFE_TIME_HousekeepingTlm_Payload_t::SecondsMET

Telemetry Mnemonic(s) **$sc_$cpu_TIME_METSecs**

Definition at line 948 of file cfe_time_msg.h.

12.177.2.9 SecondsSTCF

uint32 CFE_TIME_HousekeepingTlm_Payload_t::SecondsSTCF

Telemetry Mnemonic(s) **$sc_$cpu_TIME_STCFSecs**

Definition at line 953 of file cfe_time_msg.h.

12.177.2.10 Subsecs1HzAdj

uint32 CFE_TIME_HousekeepingTlm_Payload_t::Subsecs1HzAdj

Telemetry Mnemonic(s) **$sc_$cpu_TIME_1HzAdjSSecs**

Definition at line 964 of file cfe_time_msg.h.

12.177.2.11 SubsecsDelay

uint32 CFE_TIME_HousekeepingTlm_Payload_t::SubsecsDelay

Telemetry Mnemonic(s) **$sc_$cpu_TIME_1HzAdjSSecs**

Definition at line 974 of file cfe_time_msg.h.

12.177.2.12 SubsecsMET

```
uint32 CFE_TIME_HousekeepingTlm_Payload_t::SubsecsMET
```

Telemetry Mnemonic(s) **`$sc_$cpu_TIME_METSubsecs`**

Definition at line 950 of file cfe_time_msg.h.

12.177.2.13 SubsecsSTCF

```
uint32 CFE_TIME_HousekeepingTlm_Payload_t::SubsecsSTCF
```

Telemetry Mnemonic(s) **`$sc_$cpu_TIME_STCFSubsecs`**

Definition at line 955 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.178 CFE_TIME_HousekeepingTlm_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 TlmHeader [CFE_SB_TLM_HDR_SIZE]
- CFE_TIME_HousekeepingTlm_Payload_t Payload

12.178.1 Detailed Description

Definition at line 980 of file cfe_time_msg.h.

12.178.2 Field Documentation

12.178.2.1 Payload

```
CFE_TIME_HousekeepingTlm_Payload_t CFE_TIME_HousekeepingTlm_t::Payload
```

Definition at line 983 of file cfe_time_msg.h.

12.178.2.2 TlmHeader

```
uint8 CFE_TIME_HousekeepingTlm_t::TlmHeader[CFE_SB_TLM_HDR_SIZE]
```

Definition at line 982 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.179 CFE_TIME_LeapsCmd_Payload_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- int16 LeapSeconds

12.179.1 Detailed Description

Definition at line 749 of file cfe_time_msg.h.

12.179.2 Field Documentation

12.179.2.1 LeapSeconds

```
int16 CFE_TIME_LeapsCmd_Payload_t::LeapSeconds
```

Definition at line 751 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.180 CFE_TIME_NoArgsCmd_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

12.180.1 Detailed Description

Definition at line 731 of file cfe_time_msg.h.

12.180.2 Field Documentation

12.180.2.1 CmdHeader

`uint8 CFE_TIME_NoArgsCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 733 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.181 CFE_TIME_OneHzAdjustmentCmd_Payload_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- uint32 Seconds
- uint32 Subseconds

12.181.1 Detailed Description

Definition at line 846 of file cfe_time_msg.h.

12.181.2 Field Documentation

12.181.2.1 Seconds

```
uint32 CFE_TIME_OneHzAdjustmentCmd_Payload_t::Seconds
```

Definition at line 848 of file cfe_time_msg.h.

12.181.2.2 Subseconds

```
uint32 CFE_TIME_OneHzAdjustmentCmd_Payload_t::Subseconds
```

Definition at line 849 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.182 CFE_TIME_OneHzAdjustmentCmd_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_OneHzAdjustmentCmd_Payload_t Payload

12.182.1 Detailed Description

Definition at line 853 of file cfe_time_msg.h.

12.182.2 Field Documentation

12.182.2.1 CmdHeader

```
uint8 CFE_TIME_OneHzAdjustmentCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 855 of file cfe_time_msg.h.

12.182.2.2 Payload

```
CFE_TIME_OneHzAdjustmentCmd_Payload_t CFE_TIME_OneHzAdjustmentCmd_t::Payload
```

Definition at line 856 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.183 CFE_TIME_ResetVars_t Struct Reference

Time related variables that are maintained through a Processor Reset.

```
#include <cfe_time.h>
```

Data Fields

- uint32 Signature
  Data validation signature used to verify data structure contents.
- int16 LeapSeconds
  Leap seconds value.
- uint16 ClockSignal
  Current clock signal selection.
- CFE_TIME_SysTime_t CurrentMET
  Current Mission Elapsed Time (MET)
- CFE_TIME_SysTime_t CurrentSTCF
  Current Spacecraft Time Correlation Factor (STCF)
- CFE_TIME_SysTime_t CurrentDelay
  Current time client delay value.

12.183.1 Detailed Description

Description

The CFE_TIME_ResetVars_t data structure contains those variables that are maintained in an area of memory that is not cleared during a Processor Reset. This allows the cFE Time Service to maintain time to the best of its ability after a Processor Reset.

Definition at line 156 of file cfe_time.h.

12.183.2 Field Documentation

12.183.2.1 ClockSignal

```
uint16 CFE_TIME_ResetVars_t::ClockSignal
```

Definition at line 160 of file cfe_time.h.

12.183.2.2 CurrentDelay

```
CFE_TIME_SysTime_t CFE_TIME_ResetVars_t::CurrentDelay
```

Definition at line 163 of file cfe_time.h.

12.183.2.3 CurrentMET

```
CFE_TIME_SysTime_t CFE_TIME_ResetVars_t::CurrentMET
```

Definition at line 161 of file cfe_time.h.

12.183.2.4 CurrentSTCF

```
CFE_TIME_SysTime_t CFE_TIME_ResetVars_t::CurrentSTCF
```

Definition at line 162 of file cfe_time.h.

12.183.2.5 LeapSeconds

```
int16 CFE_TIME_ResetVars_t::LeapSeconds
```

Definition at line 159 of file cfe_time.h.

12.183.2.6 Signature

```
uint32 CFE_TIME_ResetVars_t::Signature
```

Definition at line 158 of file cfe_time.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time.h

12.184 CFE_TIME_SetLeapSeconds_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_LeapsCmd_Payload_t Payload

12.184.1 Detailed Description

Definition at line 754 of file cfe_time_msg.h.

12.184.2 Field Documentation

12.184.2.1 CmdHeader

`uint8 CFE_TIME_SetLeapSeconds_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]`

Definition at line 756 of file cfe_time_msg.h.

12.184.2.2 Payload

`CFE_TIME_LeapsCmd_Payload_t CFE_TIME_SetLeapSeconds_t::Payload`

Definition at line 757 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.185 CFE_TIME_SetSignal_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_SignalCmd_Payload_t Payload

12.185.1 Detailed Description

Definition at line 806 of file cfe_time_msg.h.

12.185.2 Field Documentation

12.185.2.1 CmdHeader

```
uint8 CFE_TIME_SetSignal_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 808 of file cfe_time_msg.h.

12.185.2.2 Payload

```
CFE_TIME_SignalCmd_Payload_t CFE_TIME_SetSignal_t::Payload
```

Definition at line 809 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.186 CFE_TIME_SetSource_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_SourceCmd_Payload_t Payload

12.186.1 Detailed Description

Definition at line 789 of file cfe_time_msg.h.

12.186.2 Field Documentation

12.186.2.1 CmdHeader

```
uint8 CFE_TIME_SetSource_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 791 of file cfe_time_msg.h.

12.186.2.2 Payload

```
CFE_TIME_SourceCmd_Payload_t CFE_TIME_SetSource_t::Payload
```

Definition at line 792 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.187 CFE_TIME_SetState_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_StateCmd_Payload_t Payload

12.187.1 Detailed Description

Definition at line 772 of file cfe_time_msg.h.

12.187.2 Field Documentation

12.187.2.1 CmdHeader

```
uint8 CFE_TIME_SetState_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 774 of file cfe_time_msg.h.

12.187.2.2 Payload

```
CFE_TIME_StateCmd_Payload_t CFE_TIME_SetState_t::Payload
```

Definition at line 775 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.188 CFE_TIME_SignalCmd_Payload_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- int16 ToneSource

  CFE_TIME_ToneSignalSelect_PRIMARY=Primary Source, CFE_TIME_ToneSignalSelect_REDUNDANT=Redundant Source

12.188.1 Detailed Description

Definition at line 799 of file cfe_time_msg.h.

12.188.2 Field Documentation

12.188.2.1 ToneSource

```
int16 CFE_TIME_SignalCmd_Payload_t::ToneSource
```

Selects either the "Primary" or "Redundant" tone signal source

Definition at line 801 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.189 CFE_TIME_SourceCmd_Payload_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- int16 TimeSource

  CFE_TIME_SourceSelect_INTERNAL=Internal Source, *CFE_TIME_SourceSelect_EXTERNAL*=External Source

12.189.1 Detailed Description

Definition at line 782 of file cfe_time_msg.h.

12.189.2 Field Documentation

12.189.2.1 TimeSource

```
int16 CFE_TIME_SourceCmd_Payload_t::TimeSource
```

Selects either the "Internal" and "External" clock source

Definition at line 784 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.190 CFE_TIME_StateCmd_Payload_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- int16 ClockState

  CFE_TIME_ClockState_INVALID=Spacecraft time has not been accurately set, *CFE_TIME_ClockState_VALID*=Spacecraft clock has been accurately set, *CFE_TIME_ClockState_FLYWHEEL*=Force into FLYWHEEL mode

12.190.1 Detailed Description

Definition at line 764 of file cfe_time_msg.h.

12.190.2 Field Documentation

12.190.2.1 ClockState

`int16 CFE_TIME_StateCmd_Payload_t::ClockState`

Selects the current clock state

Definition at line 766 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.191 CFE_TIME_SysTime_t Struct Reference

Data structure used to hold system time values.

`#include <cfe_time.h>`

Data Fields

- uint32 Seconds

  Number of seconds since epoch.
- uint32 Subseconds

  Number of subseconds since epoch (LSB = 2^(-32) seconds)

12.191.1 Detailed Description

Description

The CFE_TIME_SysTime_t data structure is used to hold time values. Time is referred to as the elapsed time (in seconds and subseconds) since a specified epoch time. The subseconds field contains the number of 2^(-32) second intervals that have elapsed since the epoch.

Definition at line 114 of file cfe_time.h.

12.191.2 Field Documentation

12.191.2.1 Seconds

`uint32 CFE_TIME_SysTime_t::Seconds`

Definition at line 116 of file cfe_time.h.

Referenced by CFE_SB_GetMsgTime(), and CFE_SB_SetMsgTime().

12.191.2.2 Subseconds

```
uint32 CFE_TIME_SysTime_t::Subseconds
```

Definition at line 117 of file cfe_time.h.

Referenced by CFE_SB_GetMsgTime(), and CFE_SB_SetMsgTime().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time.h

12.192 CFE_TIME_TimeCmd_Payload_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint32 Seconds
- uint32 MicroSeconds

12.192.1 Detailed Description

Definition at line 817 of file cfe_time_msg.h.

12.192.2 Field Documentation

12.192.2.1 MicroSeconds

```
uint32 CFE_TIME_TimeCmd_Payload_t::MicroSeconds
```

Definition at line 820 of file cfe_time_msg.h.

12.192.2.2 Seconds

```
uint32 CFE_TIME_TimeCmd_Payload_t::Seconds
```

Definition at line 819 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.193 CFE_TIME_TimeCmd_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_TimeCmd_Payload_t Payload

12.193.1 Detailed Description

Definition at line 823 of file cfe_time_msg.h.

12.193.2 Field Documentation

12.193.2.1 CmdHeader

```
uint8 CFE_TIME_TimeCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 825 of file cfe_time_msg.h.

12.193.2.2 Payload

```
CFE_TIME_TimeCmd_Payload_t CFE_TIME_TimeCmd_t::Payload
```

Definition at line 826 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.194 CFE_TIME_ToneDataCmd_Payload_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- CFE_TIME_SysTime_t AtToneMET

  MET at time of tone.
- CFE_TIME_SysTime_t AtToneSTCF

  STCF at time of tone.
- int16 AtToneLeapSeconds

  Leap Seconds at time of tone.
- int16 AtToneState

  Clock state at time of tone.

12.194.1 Detailed Description

Definition at line 901 of file cfe_time_msg.h.

12.194.2 Field Documentation

12.194.2.1 AtToneLeapSeconds

```
int16 CFE_TIME_ToneDataCmd_Payload_t::AtToneLeapSeconds
```

Definition at line 905 of file cfe_time_msg.h.

12.194.2.2 AtToneMET

```
CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload_t::AtToneMET
```

Definition at line 903 of file cfe_time_msg.h.

12.194.2.3 AtToneState

```
int16 CFE_TIME_ToneDataCmd_Payload_t::AtToneState
```

Definition at line 906 of file cfe_time_msg.h.

12.194.2.4 AtToneSTCF

```
CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload_t::AtToneSTCF
```

Definition at line 904 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.195 CFE_TIME_ToneDataCmd_t Struct Reference

```
#include <cfe_time_msg.h>
```

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]
- CFE_TIME_ToneDataCmd_Payload_t Payload

12.195.1 Detailed Description

Definition at line 909 of file cfe_time_msg.h.

12.195.2 Field Documentation

12.195.2.1 CmdHeader

```
uint8 CFE_TIME_ToneDataCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 911 of file cfe_time_msg.h.

12.195.2.2 Payload

```
CFE_TIME_ToneDataCmd_Payload_t CFE_TIME_ToneDataCmd_t::Payload
```

Definition at line 912 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.196 CFE_TIME_ToneSignalCmd_t Struct Reference

`#include <cfe_time_msg.h>`

Data Fields

- uint8 CmdHeader [CFE_SB_CMD_HDR_SIZE]

12.196.1 Detailed Description

Definition at line 881 of file cfe_time_msg.h.

12.196.2 Field Documentation

12.196.2.1 CmdHeader

```
uint8 CFE_TIME_ToneSignalCmd_t::CmdHeader[CFE_SB_CMD_HDR_SIZE]
```

Definition at line 883 of file cfe_time_msg.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/inc/cfe_time_msg.h

12.197 EVS_AppData_t Struct Reference

`#include <cfe_evs_task.h>`

Data Fields

- EVS_BinFilter_t BinFilters [CFE_PLATFORM_EVS_MAX_EVENT_FILTERS]
- uint8 ActiveFlag
- uint8 EventTypesActiveFlag
- uint16 EventCount
- uint16 RegisterFlag

12.197.1 Detailed Description

Definition at line 90 of file cfe_evs_task.h.

12.197.2 Field Documentation

12.197.2.1 ActiveFlag

```
uint8 EVS_AppData_t::ActiveFlag
```

Definition at line 94 of file cfe_evs_task.h.

Referenced by CFE_EVS_DisableAppEventsCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_Register(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_WriteAppDataFileCmd(), and EVS_IsFiltered().

12.197.2.2 BinFilters

```
EVS_BinFilter_t EVS_AppData_t::BinFilters[CFE_PLATFORM_EVS_MAX_EVENT_FILTERS]
```

Definition at line 92 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_WriteAppDataFileCmd(), and EVS_IsFiltered().

12.197.2.3 EventCount

```
uint16 EVS_AppData_t::EventCount
```

Definition at line 96 of file cfe_evs_task.h.

Referenced by CFE_EVS_Register(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_WriteAppDataFileCmd(), EVS_GenerateEventTelemetry(), and EVS_NotRegistered().

12.197.2.4 EventTypesActiveFlag

```
uint8 EVS_AppData_t::EventTypesActiveFlag
```

Definition at line 95 of file cfe_evs_task.h.

Referenced by CFE_EVS_Register(), CFE_EVS_WriteAppDataFileCmd(), EVS_DisableTypes(), EVS_EnableTypes(), and EVS_IsFiltered().

12.197.2.5 RegisterFlag

```
uint16 EVS_AppData_t::RegisterFlag
```

Definition at line 97 of file cfe_evs_task.h.

Referenced by CFE_EVS_CleanUpApp(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_EnableEventTypeCmd(), CFE_EVS_Register(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetFilter(), CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), CFE_EVS_Unregister(), CFE_EVS_WriteAppDataFileCmd(), and EVS_GetApplicationInfo().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/evs/cfe_evs_task.h

12.198 EVS_BinFilter_t Struct Reference

```
#include <cfe_evs_task.h>
```

Data Fields

- int16 EventID
- uint16 Mask
- uint16 Count
- uint16 Padding

12.198.1 Detailed Description

Definition at line 80 of file cfe_evs_task.h.

12.198.2 Field Documentation

12.198.2.1 Count

```
uint16 EVS_BinFilter_t::Count
```

Definition at line 84 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_Register(), CFE_E↩VS_ResetAllFilters(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), and E↩VS_IsFiltered().

12.198.2.2 EventID

```
int16 EVS_BinFilter_t::EventID
```

Definition at line 82 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), and CFE_EVS_Register().

12.198.2.3 Mask

```
uint16 EVS_BinFilter_t::Mask
```

Definition at line 83 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_Register(), CFE_E↩VS_SetFilterCmd(), and EVS_IsFiltered().

12.198.2.4 Padding

```
uint16 EVS_BinFilter_t::Padding
```

Definition at line 85 of file cfe_evs_task.h.

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/evs/cfe_evs_task.h

12.199 MemPoolAddr_t Union Reference

Data Fields

- BD_t * BdPtr
- uint32 * UserPtr
- cpuaddr Addr

12.199.1 Detailed Description

Union to assist/simplify the pointer manipulation when allocating buffers in a pool

When allocating buffers, the memory is calculated using raw addresses (cpuaddr) and then used as either buffer descriptor pointer (BD_t*) or user buffer pointers (uint32*).

This union assists with casting between the 3 types. It is still a cast, but at least it limits the casting to these intended data types so it is slightly safer in that regard.

Definition at line 63 of file cfe_esmempool.c.

12.199.2 Field Documentation

12.199.2.1 Addr

```
cpuaddr MemPoolAddr_t::Addr
```

Use when interpreting pool memory as a memory address

Definition at line 67 of file cfe_esmempool.c.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

12.199.2.2 BdPtr

```
BD_t* MemPoolAddr_t::BdPtr
```

Use when interpreting pool memory as a descriptor

Definition at line 65 of file cfe_esmempool.c.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

12.199.2.3 UserPtr

```
uint32* MemPoolAddr_t::UserPtr
```

Use when interpreting pool memory as a user buffer

Definition at line 66 of file cfe_esmempool.c.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

The documentation for this union was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_esmempool.c

12.200 OS_bin_sem_prop_t Struct Reference

```
#include <osapi-os-core.h>
```

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator
- int32 value

12.200.1 Detailed Description

Definition at line 73 of file osapi-os-core.h.

12.200.2 Field Documentation

12.200.2.1 creator

```
uint32 OS_bin_sem_prop_t::creator
```

Definition at line 76 of file osapi-os-core.h.

12.200.2.2 name

```
char OS_bin_sem_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 75 of file osapi-os-core.h.

12.200.2.3 value

```
int32 OS_bin_sem_prop_t::value
```

Definition at line 77 of file osapi-os-core.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.201 OS_count_sem_prop_t Struct Reference

```
#include <osapi-os-core.h>
```

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator
- int32 value

12.201.1 Detailed Description

Definition at line 81 of file osapi-os-core.h.

12.201.2 Field Documentation

12.201.2.1 creator

```
uint32 OS_count_sem_prop_t::creator
```

Definition at line 84 of file osapi-os-core.h.

12.201.2.2 name

```
char OS_count_sem_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 83 of file osapi-os-core.h.

12.201.2.3 value

```
int32 OS_count_sem_prop_t::value
```

Definition at line 85 of file osapi-os-core.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.202 os_dirent_t Struct Reference

```
#include <osapi-os-filesys.h>
```

Data Fields

- char FileName [OS_MAX_PATH_LEN]

12.202.1 Detailed Description

Definition at line 165 of file osapi-os-filesys.h.

12.202.2 Field Documentation

12.202.2.1 FileName

```
char os_dirent_t::FileName[OS_MAX_PATH_LEN]
```

Definition at line 167 of file osapi-os-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-filesys.h

12.203 OS_FdSet Struct Reference

An abstract structure capable of holding several OSAL IDs.

```
#include <osapi-os-core.h>
```

Data Fields

- uint8 object_ids [(OS_MAX_NUM_OPEN_FILES+7)/8]

12.203.1 Detailed Description

This is part of the select API and is manipulated using the related API calls. It should not be modified directly by applications.

See also

OS_SelectFdZero(), OS_SelectFdAdd(), OS_SelectFdClear(), OS_SelectFdIsSet()

Definition at line 1193 of file osapi-os-core.h.

12.203.2 Field Documentation

12.203.2.1 object_ids

```
uint8 OS_FdSet::object_ids[(OS_MAX_NUM_OPEN_FILES+7)/8]
```

Definition at line 1195 of file osapi-os-core.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.204 OS_file_prop_t Struct Reference

```
#include <osapi-os-filesys.h>
```

Data Fields

- char Path [OS_MAX_PATH_LEN]
- uint32 User
- uint8 IsValid

12.204.1 Detailed Description

Definition at line 121 of file osapi-os-filesys.h.

12.204.2 Field Documentation

12.204.2.1 IsValid

```
uint8 OS_file_prop_t::IsValid
```

Definition at line 125 of file osapi-os-filesys.h.

12.204.2.2 Path

```
char OS_file_prop_t::Path[OS_MAX_PATH_LEN]
```

Definition at line 123 of file osapi-os-filesys.h.

12.204.2.3 User

```
uint32 OS_file_prop_t::User
```

Definition at line 124 of file osapi-os-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-filesys.h

12.205 os_fsinfo_t Struct Reference

```
#include <osapi-os-filesys.h>
```

Data Fields

- uint32 MaxFds
- uint32 FreeFds
- uint32 MaxVolumes
- uint32 FreeVolumes

12.205.1 Detailed Description

Definition at line 112 of file osapi-os-filesys.h.

12.205.2 Field Documentation

12.205.2.1 FreeFds

```
uint32 os_fsinfo_t::FreeFds
```

Definition at line 115 of file osapi-os-filesys.h.

12.205.2.2 FreeVolumes

```
uint32 os_fsinfo_t::FreeVolumes
```

Definition at line 117 of file osapi-os-filesys.h.

12.205.2.3 MaxFds

```
uint32 os_fsinfo_t::MaxFds
```

Definition at line 114 of file osapi-os-filesys.h.

12.205.2.4 MaxVolumes

```
uint32 os_fsinfo_t::MaxVolumes
```

Definition at line 116 of file osapi-os-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-filesys.h

12.206 os_fstat_t Struct Reference

```
#include <osapi-os-filesys.h>
```

Data Fields

- uint32 FileModeBits
- int32 FileTime
- uint32 FileSize

12.206.1 Detailed Description

Definition at line 136 of file osapi-os-filesys.h.

12.206.2 Field Documentation

12.206.2.1 FileModeBits

```
uint32 os_fstat_t::FileModeBits
```

Definition at line 138 of file osapi-os-filesys.h.

12.206.2.2 FileSize

```
uint32 os_fstat_t::FileSize
```

Definition at line 140 of file osapi-os-filesys.h.

12.206.2.3 FileTime

```
int32 os_fstat_t::FileTime
```

Definition at line 139 of file osapi-os-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-filesys.h

12.207 OS_heap_prop_t Struct Reference

```
#include <osapi-os-core.h>
```

Data Fields

- uint32 free_bytes
- uint32 free_blocks
- uint32 largest_free_block

12.207.1 Detailed Description

Definition at line 105 of file osapi-os-core.h.

12.207.2 Field Documentation

12.207.2.1 free_blocks

```
uint32 OS_heap_prop_t::free_blocks
```

Definition at line 108 of file osapi-os-core.h.

Referenced by CFE_ES_HousekeepingCmd().

12.207.2.2 free_bytes

```
uint32 OS_heap_prop_t::free_bytes
```

Definition at line 107 of file osapi-os-core.h.

Referenced by CFE_ES_HousekeepingCmd().

12.207.2.3 largest_free_block

```
uint32 OS_heap_prop_t::largest_free_block
```

Definition at line 109 of file osapi-os-core.h.

Referenced by CFE_ES_HousekeepingCmd().

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.208 OS_module_address_t Struct Reference

```
#include <osapi-os-loader.h>
```

Data Fields

- uint32 valid
- uint32 flags
- cpuaddr code_address
- cpuaddr code_size
- cpuaddr data_address
- cpuaddr data_size
- cpuaddr bss_address
- cpuaddr bss_size

12.208.1 Detailed Description

Definition at line 31 of file osapi-os-loader.h.

12.208.2 Field Documentation

12.208.2.1 bss_address

`cpuaddr OS_module_address_t::bss_address`

Definition at line 39 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.208.2.2 bss_size

`cpuaddr OS_module_address_t::bss_size`

Definition at line 40 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.208.2.3 code_address

`cpuaddr OS_module_address_t::code_address`

Definition at line 35 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.208.2.4 code_size

`cpuaddr OS_module_address_t::code_size`

Definition at line 36 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.208.2.5 data_address

```
cpuaddr OS_module_address_t::data_address
```

Definition at line 37 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.208.2.6 data_size

```
cpuaddr OS_module_address_t::data_size
```

Definition at line 38 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.208.2.7 flags

```
uint32 OS_module_address_t::flags
```

Definition at line 34 of file osapi-os-loader.h.

12.208.2.8 valid

```
uint32 OS_module_address_t::valid
```

Definition at line 33 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-loader.h

12.209 OS_module_prop_t Struct Reference

```
#include <osapi-os-loader.h>
```

Data Fields

- cpuaddr entry_point
- cpuaddr host_module_id
- char filename [OS_MAX_PATH_LEN]
- char name [OS_MAX_API_NAME]
- OS_module_address_t addr

12.209.1 Detailed Description

Definition at line 43 of file osapi-os-loader.h.

12.209.2 Field Documentation

12.209.2.1 addr

```
OS_module_address_t OS_module_prop_t::addr
```

Definition at line 49 of file osapi-os-loader.h.

Referenced by CFE_ES_GetAppInfoInternal().

12.209.2.2 entry_point

```
cpuaddr OS_module_prop_t::entry_point
```

Definition at line 45 of file osapi-os-loader.h.

12.209.2.3 filename

```
char OS_module_prop_t::filename[OS_MAX_PATH_LEN]
```

Definition at line 47 of file osapi-os-loader.h.

12.209.2.4 host_module_id

```
cpuaddr OS_module_prop_t::host_module_id
```

Definition at line 46 of file osapi-os-loader.h.

12.209.2.5 name

```
char OS_module_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 48 of file osapi-os-loader.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-loader.h

12.210 OS_mut_sem_prop_t Struct Reference

```
#include <osapi-os-core.h>
```

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator

12.210.1 Detailed Description

Definition at line 89 of file osapi-os-core.h.

12.210.2 Field Documentation

12.210.2.1 creator

```
uint32 OS_mut_sem_prop_t::creator
```

Definition at line 92 of file osapi-os-core.h.

12.210.2.2 name

```
char OS_mut_sem_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 91 of file osapi-os-core.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.211 OS_queue_prop_t Struct Reference

`#include <osapi-os-core.h>`

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator

12.211.1 Detailed Description

Definition at line 66 of file osapi-os-core.h.

12.211.2 Field Documentation

12.211.2.1 creator

`uint32 OS_queue_prop_t::creator`

Definition at line 69 of file osapi-os-core.h.

12.211.2.2 name

`char OS_queue_prop_t::name[OS_MAX_API_NAME]`

Definition at line 68 of file osapi-os-core.h.

Referenced by CFE_SB_GetPipeName().

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.212 OS_SockAddr_t Struct Reference

`#include <osapi-os-net.h>`

Data Fields

- uint32 ActualLength
- OS_SockAddrData_t AddrData

12.212.1 Detailed Description

Encapsulates a generic network address

This is just an abstract buffer type that holds a network address. It is allocated for the worst-case size defined by OS_SOCKADDR_MAX_LEN, and the real size is stored within.

Definition at line 90 of file osapi-os-net.h.

12.212.2 Field Documentation

12.212.2.1 ActualLength

```
uint32 OS_SockAddr_t::ActualLength
```

Length of the actual address data

Definition at line 92 of file osapi-os-net.h.

12.212.2.2 AddrData

```
OS_SockAddrData_t OS_SockAddr_t::AddrData
```

Abstract Address data

Definition at line 93 of file osapi-os-net.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-net.h

12.213 OS_SockAddrData_t Union Reference

```
#include <osapi-os-net.h>
```

Data Fields

- uint8 Buffer [OS_SOCKADDR_MAX_LEN]
- uint32 AlignU32
- void * AlignPtr

12.213.1 Detailed Description

Storage buffer for generic network address

This is a union type that helps to ensure a minimum alignment value for the data storage, such that it can be cast to the system-specific type without increasing alignment requirements.

Definition at line 76 of file osapi-os-net.h.

12.213.2 Field Documentation

12.213.2.1 AlignPtr

```
void* OS_SockAddrData_t::AlignPtr
```

Ensures pointer alignment

Definition at line 80 of file osapi-os-net.h.

12.213.2.2 AlignU32

```
uint32 OS_SockAddrData_t::AlignU32
```

Ensures uint32 alignment

Definition at line 79 of file osapi-os-net.h.

12.213.2.3 Buffer

```
uint8 OS_SockAddrData_t::Buffer[OS_SOCKADDR_MAX_LEN]
```

Ensures length of at least OS_SOCKADDR_MAX_LEN

Definition at line 78 of file osapi-os-net.h.

The documentation for this union was generated from the following file:

- osal/src/os/inc/osapi-os-net.h

12.214 OS_socket_prop_t Struct Reference

```
#include <osapi-os-net.h>
```

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator

12.214.1 Detailed Description

Encapsulates socket properties

This is for consistency with other OSAL resource types. Currently no extra properties are exposed here but this could change in a future revision of OSAL as needed.

Definition at line 103 of file osapi-os-net.h.

12.214.2 Field Documentation

12.214.2.1 creator

```
uint32 OS_socket_prop_t::creator
```

OSAL TaskID which opened the socket

Definition at line 106 of file osapi-os-net.h.

12.214.2.2 name

```
char OS_socket_prop_t::name[OS_MAX_API_NAME]
```

Name of the socket

Definition at line 105 of file osapi-os-net.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-net.h

12.215 OS_static_symbol_record_t Struct Reference

```
#include <osapi-os-loader.h>
```

Data Fields

- const char * Name
- void(* Address)(void)
- const char * Module

12.215.1 Detailed Description

Associates a single symbol name with a memory address.

If the OS_STATIC_SYMBOL_TABLE feature is enabled, then an array of these structures should be provided by the application. When the application needs to find a symbol address, the static table will be checked in addition to (or instead of) the OS/library-provided lookup function.

This static symbol allows systems that do not implement dynamic module loading to maintain the same semantics as dynamically loaded modules.

Definition at line 65 of file osapi-os-loader.h.

12.215.2 Field Documentation

12.215.2.1 Address

```
void(* OS_static_symbol_record_t::Address) (void)
```

Definition at line 68 of file osapi-os-loader.h.

12.215.2.2 Module

```
const char* OS_static_symbol_record_t::Module
```

Definition at line 69 of file osapi-os-loader.h.

12.215.2.3 Name

```
const char* OS_static_symbol_record_t::Name
```

Definition at line 67 of file osapi-os-loader.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-loader.h

12.216 OS_task_prop_t Struct Reference

`#include <osapi-os-core.h>`

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator
- uint32 stack_size
- uint32 priority
- uint32 OStask_id

12.216.1 Detailed Description

Definition at line 56 of file osapi-os-core.h.

12.216.2 Field Documentation

12.216.2.1 creator

`uint32 OS_task_prop_t::creator`

Definition at line 59 of file osapi-os-core.h.

12.216.2.2 name

`char OS_task_prop_t::name[OS_MAX_API_NAME]`

Definition at line 58 of file osapi-os-core.h.

12.216.2.3 OStask_id

`uint32 OS_task_prop_t::OStask_id`

Definition at line 62 of file osapi-os-core.h.

Referenced by CFE_ES_ProcessCoreException().

12.216.2.4 priority

```
uint32 OS_task_prop_t::priority
```

Definition at line 61 of file osapi-os-core.h.

12.216.2.5 stack_size

```
uint32 OS_task_prop_t::stack_size
```

Definition at line 60 of file osapi-os-core.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.217 OS_time_t Struct Reference

```
#include <osapi-os-core.h>
```

Data Fields

- uint32 seconds
- uint32 microsecs

12.217.1 Detailed Description

Definition at line 98 of file osapi-os-core.h.

12.217.2 Field Documentation

12.217.2.1 microsecs

```
uint32 OS_time_t::microsecs
```

Definition at line 101 of file osapi-os-core.h.

Referenced by CFE_PSP_Get_Timebase().

12.217.2.2 seconds

```
uint32 OS_time_t::seconds
```

Definition at line 100 of file osapi-os-core.h.

Referenced by CFE_PSP_Get_Timebase().

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-core.h

12.218 OS_timebase_prop_t Struct Reference

```
#include <osapi-os-timer.h>
```

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator
- uint32 nominal_interval_time
- uint32 freerun_time
- uint32 accuracy

12.218.1 Detailed Description

Definition at line 38 of file osapi-os-timer.h.

12.218.2 Field Documentation

12.218.2.1 accuracy

```
uint32 OS_timebase_prop_t::accuracy
```

Definition at line 44 of file osapi-os-timer.h.

12.218.2.2 creator

```
uint32 OS_timebase_prop_t::creator
```

Definition at line 41 of file osapi-os-timer.h.

12.218.2.3 freerun_time

```
uint32 OS_timebase_prop_t::freerun_time
```

Definition at line 43 of file osapi-os-timer.h.

12.218.2.4 name

```
char OS_timebase_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 40 of file osapi-os-timer.h.

12.218.2.5 nominal_interval_time

```
uint32 OS_timebase_prop_t::nominal_interval_time
```

Definition at line 42 of file osapi-os-timer.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-timer.h

12.219 OS_timer_prop_t Struct Reference

```
#include <osapi-os-timer.h>
```

Data Fields

- char name [OS_MAX_API_NAME]
- uint32 creator
- uint32 start_time
- uint32 interval_time
- uint32 accuracy

12.219.1 Detailed Description

Definition at line 28 of file osapi-os-timer.h.

12.219.2 Field Documentation

12.219.2.1 accuracy

```
uint32 OS_timer_prop_t::accuracy
```

Definition at line 34 of file osapi-os-timer.h.

12.219.2.2 creator

```
uint32 OS_timer_prop_t::creator
```

Definition at line 31 of file osapi-os-timer.h.

12.219.2.3 interval_time

```
uint32 OS_timer_prop_t::interval_time
```

Definition at line 33 of file osapi-os-timer.h.

12.219.2.4 name

```
char OS_timer_prop_t::name[OS_MAX_API_NAME]
```

Definition at line 30 of file osapi-os-timer.h.

12.219.2.5 start_time

```
uint32 OS_timer_prop_t::start_time
```

Definition at line 32 of file osapi-os-timer.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-timer.h

12.220 OS_VolumeInfo_t Struct Reference

```
#include <osapi-os-filesys.h>
```

Data Fields

- char DeviceName [OS_FS_DEV_NAME_LEN]
- char PhysDevName [OS_FS_PHYS_NAME_LEN]
- uint32 VolumeType
- uint8 VolatileFlag
- uint8 FreeFlag
- uint8 IsMounted
- char VolumeName [OS_FS_VOL_NAME_LEN]
- char MountPoint [OS_MAX_PATH_LEN]
- uint32 BlockSize

12.220.1 Detailed Description

Definition at line 98 of file osapi-os-filesys.h.

12.220.2 Field Documentation

12.220.2.1 BlockSize

```
uint32 OS_VolumeInfo_t::BlockSize
```

Definition at line 108 of file osapi-os-filesys.h.

12.220.2.2 DeviceName

```
char OS_VolumeInfo_t::DeviceName[OS_FS_DEV_NAME_LEN]
```

Definition at line 100 of file osapi-os-filesys.h.

12.220.2.3 FreeFlag

```
uint8 OS_VolumeInfo_t::FreeFlag
```

Definition at line 104 of file osapi-os-filesys.h.

12.220.2.4 IsMounted

```
uint8 OS_VolumeInfo_t::IsMounted
```

Definition at line 105 of file osapi-os-filesys.h.

12.220.2.5 MountPoint

```
char OS_VolumeInfo_t::MountPoint[OS_MAX_PATH_LEN]
```

Definition at line 107 of file osapi-os-filesys.h.

12.220.2.6 PhysDevName

```
char OS_VolumeInfo_t::PhysDevName[OS_FS_PHYS_NAME_LEN]
```

Definition at line 101 of file osapi-os-filesys.h.

12.220.2.7 VolatileFlag

```
uint8 OS_VolumeInfo_t::VolatileFlag
```

Definition at line 103 of file osapi-os-filesys.h.

12.220.2.8 VolumeName

```
char OS_VolumeInfo_t::VolumeName[OS_FS_VOL_NAME_LEN]
```

Definition at line 106 of file osapi-os-filesys.h.

12.220.2.9 VolumeType

```
uint32 OS_VolumeInfo_t::VolumeType
```

Definition at line 102 of file osapi-os-filesys.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/osapi-os-filesys.h

12.221 Pool_t Struct Reference

`#include <cfe_esmempool.h>`

Data Fields

- cpuaddr PoolHandle
- cpuaddr Size
- cpuaddr End
- cpuaddr CurrentAddr
- cpuaddr AlignMask
- BlockSizeDesc_t * SizeDescPtr
- uint16 CheckErrCntr
- uint16 RequestCntr
- uint32 MutexId
- uint32 UseMutex
- BlockSizeDesc_t SizeDesc [CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]

12.221.1 Detailed Description

Definition at line 61 of file cfe_esmempool.h.

12.221.2 Field Documentation

12.221.2.1 AlignMask

`cpuaddr Pool_t::AlignMask`

Definition at line 67 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), and CFE_ES_PoolCreateEx().

12.221.2.2 CheckErrCntr

`uint16 Pool_t::CheckErrCntr`

Definition at line 69 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_PoolCreateEx(), and CFE_ES_PutPoolBuf().

12.221.2.3 CurrentAddr

```
cpuaddr Pool_t::CurrentAddr
```

Definition at line 66 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), and CFE_ES_PoolCreateEx().

12.221.2.4 End

```
cpuaddr Pool_t::End
```

Definition at line 65 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_Pool↩CreateEx(), CFE_ES_PutPoolBuf(), and CFE_ES_ValidateHandle().

12.221.2.5 MutexId

```
uint32 Pool_t::MutexId
```

Definition at line 71 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_PoolCreateEx(), and CFE_ES_PutPool↩Buf().

12.221.2.6 PoolHandle

```
cpuaddr Pool_t::PoolHandle
```

Definition at line 63 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_Pool↩CreateEx(), CFE_ES_PutPoolBuf(), and CFE_ES_ValidateHandle().

12.221.2.7 RequestCntr

```
uint16 Pool_t::RequestCntr
```

Definition at line 70 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), and CFE_ES_PoolCreateEx().

12.221.2.8 Size

```
cpuaddr Pool_t::Size
```

Definition at line 64 of file cfe_esmempool.h.

Referenced by CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_PoolCreateEx(), and CFE_ES_↩ ValidateHandle().

12.221.2.9 SizeDesc

```
BlockSizeDesc_t Pool_t::SizeDesc[CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]
```

Definition at line 73 of file cfe_esmempool.h.

Referenced by CFE_ES_GetBlockSize(), CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_Pool↩ CreateEx(), and CFE_ES_PutPoolBuf().

12.221.2.10 SizeDescPtr

```
BlockSizeDesc_t* Pool_t::SizeDescPtr
```

Definition at line 68 of file cfe_esmempool.h.

Referenced by CFE_ES_GetBlockSize(), CFE_ES_GetPoolBuf(), CFE_ES_PoolCreateEx(), and CFE_ES_PutPool↩ Buf().

12.221.2.11 UseMutex

```
uint32 Pool_t::UseMutex
```

Definition at line 72 of file cfe_esmempool.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_PoolCreateEx(), and CFE_ES_PutPool↩ Buf().

The documentation for this struct was generated from the following file:

- cfe/fsw/cfe-core/src/es/cfe_esmempool.h

12.222 Target_PspConfigData Struct Reference

```
#include <cfe_psp_configdata.h>
```

Data Fields

- uint32 PSP_WatchdogMin
- uint32 PSP_WatchdogMax
- uint32 PSP_MemTableSize
- CFE_PSP_MemTable_t * PSP_MemoryTable
- uint32 OS_VolumeTableSize
- OS_VolumeInfo_t * OS_VolumeTable
- uint32 OS_CpuContextSize
- uint32 HW_NumEepromBanks
- CFE_PSP_VersionInfo_t PSP_VersionInfo

12.222.1 Detailed Description

PSP/Hardware configuration parameters This structure should be instantiated by the PSP according such that other modules do not need to directly include the PSP configuration at compile time.

Definition at line 56 of file cfe_psp_configdata.h.

12.222.2 Field Documentation

12.222.2.1 HW_NumEepromBanks

```
uint32 Target_PspConfigData::HW_NumEepromBanks
```

Number of EEPROM banks on this platform

Definition at line 76 of file cfe_psp_configdata.h.

12.222.2.2 OS_CpuContextSize

```
uint32 Target_PspConfigData::OS_CpuContextSize
```

Processor Context type. This is needed to determine the size of the context entry in the ER log. It is a placeholder as the implementation to use it is not merged in yet.

Definition at line 71 of file cfe_psp_configdata.h.

12.222.2.3 OS_VolumeTable

```
OS_VolumeInfo_t* Target_PspConfigData::OS_VolumeTable
```

Pointer to OS volume table (forward reference)

Definition at line 64 of file cfe_psp_configdata.h.

12.222.2.4 OS_VolumeTableSize

```
uint32 Target_PspConfigData::OS_VolumeTableSize
```

Size of OS volume table

Definition at line 63 of file cfe_psp_configdata.h.

12.222.2.5 PSP_MemoryTable

```
CFE_PSP_MemTable_t* Target_PspConfigData::PSP_MemoryTable
```

Pointer to PSP memory table (forward reference)

Definition at line 61 of file cfe_psp_configdata.h.

12.222.2.6 PSP_MemTableSize

```
uint32 Target_PspConfigData::PSP_MemTableSize
```

Size of PSP memory table

Definition at line 60 of file cfe_psp_configdata.h.

12.222.2.7 PSP_VersionInfo

```
CFE_PSP_VersionInfo_t Target_PspConfigData::PSP_VersionInfo
```

Definition at line 78 of file cfe_psp_configdata.h.

12.222.2.8 PSP_WatchdogMax

```
uint32 Target_PspConfigData::PSP_WatchdogMax
```

PSP Maximum watchdog in milliseconds

Definition at line 59 of file cfe_psp_configdata.h.

12.222.2.9 PSP_WatchdogMin

```
uint32 Target_PspConfigData::PSP_WatchdogMin
```

PSP Minimum watchdog in milliseconds

Definition at line 58 of file cfe_psp_configdata.h.

The documentation for this struct was generated from the following file:

- psp/fsw/inc/cfe_psp_configdata.h

13 File Documentation

13.1 cpu1_msgids.h File Reference

```
#include "cfe_mission_cfg.h"
```

Macros

- #define CFE_EVS_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_EVS_CMD_MSG /* 0x1801 */
- #define CFE_SB_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_SB_CMD_MSG /* 0x1803 */
- #define CFE_TBL_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TBL_CMD_MSG /* 0x1804 */
- #define CFE_TIME_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_CMD_MSG /* 0x1805 */
- #define CFE_ES_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_ES_CMD_MSG /* 0x1806 */
- #define CFE_ES_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_ES_SEND_HK_MSG /* 0x1808 */
- #define CFE_EVS_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_EVS_SEND_HK_MSG /* 0x1809 */
- #define CFE_SB_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_SB_SEND_HK_MSG /* 0x180B */
- #define CFE_TBL_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TBL_SEND_HK_MSG /* 0x180C */

- #define CFE_TIME_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_SEND_HK_MSG /* 0x180D */
- #define CFE_TIME_TONE_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_TONE_CMD_MSG /* 0x1810 */
- #define CFE_TIME_1HZ_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_1HZ_CMD_MSG /* 0x1811 */
- #define CFE_TIME_DATA_CMD_MID CFE_MISSION_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_DATA_CMD_MSG /* 0x1860 */
- #define CFE_TIME_SEND_CMD_MID CFE_MISSION_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_SEND_CMD_MSG /* 0x1862 */
- #define CFE_ES_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_HK_TLM_MSG /* 0x0800 */
- #define CFE_EVS_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_EVS_HK_TLM_MSG /* 0x0801 */
- #define CFE_SB_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_HK_TLM_MSG /* 0x0803 */
- #define CFE_TBL_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TBL_HK_TLM_MSG /* 0x0804 */
- #define CFE_TIME_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TIME_HK_TLM_MSG /* 0x0805 */
- #define CFE_TIME_DIAG_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TIME_DIAG_TLM_MSG /* 0x0806 */
- #define CFE_EVS_LONG_EVENT_MSG_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_EVS_LONG_EVENT_MSG_MSG /* 0x0808 */
- #define CFE_EVS_SHORT_EVENT_MSG_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG /* 0x0809 */
- #define CFE_SB_STATS_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_STATS_TLM_MSG /* 0x080A */
- #define CFE_ES_APP_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_APP_TLM_MSG /* 0x080B */
- #define CFE_TBL_REG_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TBL_REG_TLM_MSG /* 0x080C */
- #define CFE_SB_ALLSUBS_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_ALLSUBS_TLM_MSG /* 0x080D */
- #define CFE_SB_ONESUB_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_ONESUB_TLM_MSG /* 0x080E */
- #define CFE_ES_SHELL_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_SHELL_TLM_MSG /* 0x080F */
- #define CFE_ES_MEMSTATS_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_MEMSTATS_TLM_MSG /* 0x0810 */
- #define CFE_EVS_EVENT_MSG_MID CFE_EVS_LONG_EVENT_MSG_MID

13.1.1 Macro Definition Documentation

13.1.1.1 CFE_ES_APP_TLM_MID

```
#define CFE_ES_APP_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_APP_TLM_MSG /* 0x080B */
```

Definition at line 85 of file cpu1_msgids.h.

Referenced by CFE_ES_TaskInit().

13.1.1.2 CFE_ES_CMD_MID

```
#define CFE_ES_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_ES_CMD_MSG /* 0x1806 */
```

Definition at line 52 of file cpu1_msgids.h.

Referenced by CFE_ES_TaskInit(), and CFE_ES_TaskPipe().

13.1.1.3 CFE_ES_HK_TLM_MID

```
#define CFE_ES_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_HK_TLM_MSG /* 0x0800 */
```

Definition at line 75 of file cpu1_msgids.h.

Referenced by CFE_ES_TaskInit().

13.1.1.4 CFE_ES_MEMSTATS_TLM_MID

```
#define CFE_ES_MEMSTATS_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_MEMSTATS_TLM_MSG /*
0x0810 */
```

Definition at line 90 of file cpu1_msgids.h.

Referenced by CFE_ES_TaskInit().

13.1.1.5 CFE_ES_SEND_HK_MID

```
#define CFE_ES_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_ES_SEND_HK_MSG /* 0x1808 */
```

Definition at line 54 of file cpu1_msgids.h.

Referenced by CFE_ES_TaskInit(), and CFE_ES_TaskPipe().

13.1.1.6 CFE_ES_SHELL_TLM_MID

```
#define CFE_ES_SHELL_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_ES_SHELL_TLM_MSG /* 0x080F */
```

Definition at line 89 of file cpu1_msgids.h.

Referenced by CFE_ES_TaskInit().

13.1.1.7 CFE_EVS_CMD_MID

```
#define CFE_EVS_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_EVS_CMD_MSG /* 0x1801 */
```

Definition at line 47 of file cpu1_msgids.h.

Referenced by CFE_EVS_ProcessCommandPacket(), and CFE_EVS_TaskInit().

13.1.1.8 CFE_EVS_EVENT_MSG_MID

```
#define CFE_EVS_EVENT_MSG_MID CFE_EVS_LONG_EVENT_MSG_MID
```

Definition at line 98 of file cpu1_msgids.h.

13.1.1.9 CFE_EVS_HK_TLM_MID

```
#define CFE_EVS_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_EVS_HK_TLM_MSG /* 0x0801 */
```

Definition at line 76 of file cpu1_msgids.h.

Referenced by CFE_EVS_EarlyInit().

13.1.1.10 CFE_EVS_LONG_EVENT_MSG_MID

```
#define CFE_EVS_LONG_EVENT_MSG_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_EVS_LONG_EVENT_MSG_MSG
/* 0x0808 */
```

Definition at line 82 of file cpu1_msgids.h.

Referenced by EVS_GenerateEventTelemetry().

13.1.1.11 CFE_EVS_SEND_HK_MID

```
#define CFE_EVS_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_EVS_SEND_HK_MSG /* 0x1809 */
```

Definition at line 55 of file cpu1_msgids.h.

Referenced by CFE_EVS_ProcessCommandPacket(), and CFE_EVS_TaskInit().

13.1.1.12 CFE_EVS_SHORT_EVENT_MSG_MID

```
#define CFE_EVS_SHORT_EVENT_MSG_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_EVS_SHORT_EVENT_MSG_↩
MSG /* 0x0809 */
```

Definition at line 83 of file cpu1_msgids.h.

Referenced by EVS_GenerateEventTelemetry().

13.1.1.13 CFE_SB_ALLSUBS_TLM_MID

```
#define CFE_SB_ALLSUBS_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_ALLSUBS_TLM_MSG /* 0x080↩
D */
```

Definition at line 87 of file cpu1_msgids.h.

Referenced by CFE_SB_AppInit().

13.1.1.14 CFE_SB_CMD_MID

```
#define CFE_SB_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_SB_CMD_MSG /* 0x1803 */
```

Definition at line 49 of file cpu1_msgids.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_ProcessCmdPipePkt().

13.1.1.15 CFE_SB_HK_TLM_MID

```
#define CFE_SB_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_HK_TLM_MSG /* 0x0803 */
```

Definition at line 78 of file cpu1_msgids.h.

Referenced by CFE_SB_AppInit().

13.1.1.16 CFE_SB_ONESUB_TLM_MID

```
#define CFE_SB_ONESUB_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_ONESUB_TLM_MSG /* 0x080E
*/
```

Definition at line 88 of file cpu1_msgids.h.

Referenced by CFE_SB_AppInit().

13.1.1.17 CFE_SB_SEND_HK_MID

```
#define CFE_SB_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_SB_SEND_HK_MSG /* 0x180B */
```

Definition at line 57 of file cpu1_msgids.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_ProcessCmdPipePkt().

13.1.1.18 CFE_SB_STATS_TLM_MID

```
#define CFE_SB_STATS_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_SB_STATS_TLM_MSG /* 0x080A */
```

Definition at line 84 of file cpu1_msgids.h.

Referenced by CFE_SB_EarlyInit().

13.1.1.19 CFE_TBL_CMD_MID

```
#define CFE_TBL_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TBL_CMD_MSG /* 0x1804 */
```

Definition at line 50 of file cpu1_msgids.h.

13.1.1.20 CFE_TBL_HK_TLM_MID

```
#define CFE_TBL_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TBL_HK_TLM_MSG /* 0x0804 */
```

Definition at line 79 of file cpu1_msgids.h.

13.1.1.21 CFE_TBL_REG_TLM_MID

```
#define CFE_TBL_REG_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TBL_REG_TLM_MSG /* 0x080C */
```

Definition at line 86 of file cpu1_msgids.h.

13.1.1.22 CFE_TBL_SEND_HK_MID

```
#define CFE_TBL_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TBL_SEND_HK_MSG /* 0x180C */
```

Definition at line 58 of file cpu1_msgids.h.

13.1.1.23 CFE_TIME_1HZ_CMD_MID

```
#define CFE_TIME_1HZ_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_1HZ_CMD_MSG /* 0x1811 */
```

Definition at line 62 of file cpu1_msgids.h.

13.1.1.24 CFE_TIME_CMD_MID

```
#define CFE_TIME_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_CMD_MSG /* 0x1805 */
```

Definition at line 51 of file cpu1_msgids.h.

13.1.1.25 CFE_TIME_DATA_CMD_MID

```
#define CFE_TIME_DATA_CMD_MID CFE_MISSION_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_DATA_CMD_MSG /*
0x1860 */
```

Definition at line 68 of file cpu1_msgids.h.

13.1.1.26 CFE_TIME_DIAG_TLM_MID

```
#define CFE_TIME_DIAG_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TIME_DIAG_TLM_MSG /* 0x0806
*/
```

Definition at line 81 of file cpu1_msgids.h.

13.1.1.27 CFE_TIME_HK_TLM_MID

```
#define CFE_TIME_HK_TLM_MID CFE_MISSION_TLM_MID_BASE1 + CFE_MISSION_TIME_HK_TLM_MSG /* 0x0805 */
```

Definition at line 80 of file cpu1_msgids.h.

13.1.1.28 CFE_TIME_SEND_CMD_MID

```
#define CFE_TIME_SEND_CMD_MID CFE_MISSION_CMD_MID_BASE_GLOB + CFE_MISSION_TIME_SEND_CMD_MSG /*
0x1862 */
```

Definition at line 69 of file cpu1_msgids.h.

13.1.1.29 CFE_TIME_SEND_HK_MID

```
#define CFE_TIME_SEND_HK_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_SEND_HK_MSG /* 0x180D */
```

Definition at line 59 of file cpu1_msgids.h.

13.1.1.30 CFE_TIME_TONE_CMD_MID

```
#define CFE_TIME_TONE_CMD_MID CFE_MISSION_CMD_MID_BASE1 + CFE_MISSION_TIME_TONE_CMD_MSG /* 0x1810
*/
```

Definition at line 61 of file cpu1_msgids.h.

13.2 cpu1_platform_cfg.h File Reference

```
#include "cfe_mission_cfg.h"
```

Macros

- #define CFE_PLATFORM_CPU_ID 1
- #define CFE_PLATFORM_CPU_NAME "CPU1"
- #define CFE_PLATFORM_SB_MAX_MSG_IDS 256
- #define CFE_PLATFORM_SB_MAX_PIPES 64
- #define CFE_PLATFORM_SB_MAX_DEST_PER_PKT 16
- #define CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT 4
- #define CFE_PLATFORM_SB_BUF_MEMORY_BYTES 524288
- #define CFE_PLATFORM_SB_MAX_PIPE_DEPTH 256
- #define CFE_PLATFORM_SB_HIGHEST_VALID_MSGID 0x1FFF
- #define CFE_PLATFORM_ENDIAN CCSDS_LITTLE_ENDIAN
- #define CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME "/ram/cfe_sb_route.dat"
- #define CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME "/ram/cfe_sb_pipe.dat"
- #define CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME "/ram/cfe_sb_msgmap.dat"
- #define CFE_PLATFORM_SB_FILTERED_EVENT1 CFE_SB_SEND_NO_SUBS_EID
- #define CFE_PLATFORM_SB_FILTER_MASK1 CFE_EVS_FIRST_4_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
- #define CFE_PLATFORM_SB_FILTER_MASK2 CFE_EVS_FIRST_4_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
- #define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
- #define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
- #define CFE_PLATFORM_SB_FILTERED_EVENT5 0
- #define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_FILTERED_EVENT6 0
- #define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_FILTERED_EVENT7 0

- #define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_FILTERED_EVENT8 0
- #define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01 8
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
- #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
- #define CFE_PLATFORM_SB_MAX_BLOCK_SIZE (CFE_MISSION_SB_MAX_SB_MSG_SIZE + 40)
- #define CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER 1
- #define CFE_PLATFORM_TIME_CFG_SERVER true
- #define CFE_PLATFORM_TIME_CFG_CLIENT false
- #define CFE_PLATFORM_TIME_CFG_VIRTUAL true
- #define CFE_PLATFORM_TIME_CFG_SIGNAL false
- #define CFE_PLATFORM_TIME_CFG_SOURCE false
- #define CFE_PLATFORM_TIME_CFG_SRC_MET false
- #define CFE_PLATFORM_TIME_CFG_SRC_GPS false
- #define CFE_PLATFORM_TIME_CFG_SRC_TIME false
- #define CFE_PLATFORM_TIME_MAX_DELTA_SECS 0
- #define CFE_PLATFORM_TIME_MAX_DELTA_SUBS 500000
- #define CFE_PLATFORM_TIME_MAX_LOCAL_SECS 27
- #define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0
- #define CFE_PLATFORM_TIME_CFG_TONE_LIMIT 20000
- #define CFE_PLATFORM_TIME_CFG_START_FLY 2
- #define CFE_PLATFORM_TIME_CFG_LATCH_FLY 8
- #define CFE_PLATFORM_ES_MAX_APPLICATIONS 32
- #define CFE_PLATFORM_ES_MAX_LIBRARIES 10
- #define CFE_PLATFORM_ES_ER_LOG_ENTRIES 20
- #define CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE 128
- #define CFE_PLATFORM_ES_SYSTEM_LOG_SIZE 3072
- #define CFE_PLATFORM_ES_OBJECT_TABLE_SIZE 30
- #define CFE_PLATFORM_ES_MAX_GEN_COUNTERS 8
- #define CFE_PLATFORM_ES_APP_SCAN_RATE 1000
- #define CFE_PLATFORM_ES_APP_KILL_TIMEOUT 5
- #define CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE 512
- #define CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS 4096
- #define CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED 30
- #define CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING "/ram"
- #define CFE_PLATFORM_ES_CDS_SIZE (128 * 1024)
- #define CFE_PLATFORM_ES_USER_RESERVED_SIZE (1024 * 1024)

- #define CFE_PLATFORM_ES_RESET_AREA_SIZE (170 ∗ 1024)
- #define CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN 4
- #define CFE_PLATFORM_ES_NONVOL_STARTUP_FILE "/cf/cfe_es_startup.scr"
- #define CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE "/ram/cfe_es_startup.scr"
- #define CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME "/ram/ShellCmd.out"
- #define CFE_PLATFORM_ES_MAX_SHELL_CMD 64
- #define CFE_PLATFORM_ES_MAX_SHELL_PKT 64
- #define CFE_PLATFORM_ES_SHELL_OS_DELAY_MILLISEC 200
- #define CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE "/ram/cfe_es_app_info.log"
- #define CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE "/ram/cfe_es_task_info.log"
- #define CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE "/ram/cfe_es_syslog.log"
- #define CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE "/ram/cfe_erlog.log"
- #define CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME "/ram/cfe_es_perf.dat"
- #define CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE "/ram/cfe_cds_reg.log"
- #define CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE 1
- #define CFE_PLATFORM_ES_PERF_MAX_IDS 128
- #define CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE 10000
- #define CFE_PLATFORM_ES_PERF_FILTMASK_NONE 0
- #define CFE_PLATFORM_ES_PERF_FILTMASK_ALL ∼CFE_PLATFORM_ES_PERF_FILTMASK_NONE
- #define CFE_PLATFORM_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_PERF_FILTMASK_ALL
- #define CFE_PLATFORM_ES_PERF_TRIGMASK_NONE 0
- #define CFE_PLATFORM_ES_PERF_TRIGMASK_ALL ∼CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
- #define CFE_PLATFORM_ES_PERF_TRIGMASK_INIT CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
- #define CFE_PLATFORM_ES_PERF_CHILD_PRIORITY 200
- #define CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE 4096
- #define CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY 20
- #define CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS 50
- #define CFE_PLATFORM_ES_DEFAULT_STACK_SIZE 8192
- #define CFE_PLATFORM_ES_EXCEPTION_FUNCTION CFE_ES_ProcessCoreException
- #define CFE_PLATFORM_EVS_START_TASK_PRIORITY 61
- #define CFE_PLATFORM_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
- #define CFE_PLATFORM_SB_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_ES_START_TASK_PRIORITY 68
- #define CFE_PLATFORM_ES_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_TIME_START_TASK_PRIORITY 60
- #define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
- #define CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY 25
- #define CFE_PLATFORM_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_S↩ IZE
- #define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
- #define CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE 8192
- #define CFE_PLATFORM_TBL_START_TASK_PRIORITY 70
- #define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES 512
- #define CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS 2
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 8
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 16
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 32
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 48
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 64

- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 96
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 128
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 160
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09 256
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10 512
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11 1024
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12 2048
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13 4096
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14 8192
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15 16384
- #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16 32768
- #define CFE_PLATFORM_ES_MAX_BLOCK_SIZE 80000
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01 8
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 16
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 32
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 48
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 64
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 96
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 128
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 160
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 256
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 512
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 1024
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 2048
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 4096
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14 8192
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15 16384
- #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 32768
- #define CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE 80000
- #define CFE_PLATFORM_EVS_MAX_EVENT_FILTERS 8
- #define CFE_PLATFORM_EVS_LOG_ON
- #define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE "/ram/cfe_evs.log"
- #define CFE_PLATFORM_EVS_LOG_MAX 20
- #define CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE "/ram/cfe_evs_app.dat"
- #define CFE_PLATFORM_EVS_PORT_DEFAULT 0x0001
- #define CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG 0xE
- #define CFE_PLATFORM_EVS_DEFAULT_LOG_MODE 1
- #define CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE CFE_EVS_MsgFormat_LONG
- #define CFE_PLATFORM_TBL_BUF_MEMORY_BYTES 524288
- #define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE 16384
- #define CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE 16384
- #define CFE_PLATFORM_TBL_MAX_NUM_TABLES 128
- #define CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES 32
- #define CFE_PLATFORM_TBL_MAX_NUM_HANDLES 256
- #define CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS 4
- #define CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS 10
- #define CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE "/ram/cfe_tbl_reg.log"
- #define CFE_PLATFORM_TBL_VALID_SCID_COUNT 0
- #define CFE_PLATFORM_TBL_U32FROM4CHARS(_C1, _C2, _C3, _C4)
- #define CFE_PLATFORM_TBL_VALID_SCID_1 (CFE_MISSION_SPACECRAFT_ID)
- #define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))

- #define CFE_PLATFORM_TBL_VALID_PRID_COUNT 0
- #define CFE_PLATFORM_TBL_VALID_PRID_1 (CFE_PLATFORM_CPU_ID)
- #define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
- #define CFE_PLATFORM_TBL_VALID_PRID_3 0
- #define CFE_PLATFORM_TBL_VALID_PRID_4 0
- #define CFE_MISSION_REV 0
- #define CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC 50
- #define CFE_PLATFORM_CORE_MAX_STARTUP_MSEC 30000
- #define CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC 1000
- #define CFE_CPU_ID CFE_PLATFORM_CPU_ID
- #define CFE_CPU_NAME CFE_PLATFORM_CPU_NAME
- #define CFE_SB_MAX_MSG_IDS CFE_PLATFORM_SB_MAX_MSG_IDS
- #define CFE_SB_MAX_PIPES CFE_PLATFORM_SB_MAX_PIPES
- #define CFE_SB_MAX_DEST_PER_PKT CFE_PLATFORM_SB_MAX_DEST_PER_PKT
- #define CFE_SB_DEFAULT_MSG_LIMIT CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
- #define CFE_SB_BUF_MEMORY_BYTES CFE_PLATFORM_SB_BUF_MEMORY_BYTES
- #define CFE_SB_MAX_PIPE_DEPTH CFE_PLATFORM_SB_MAX_PIPE_DEPTH
- #define CFE_SB_HIGHEST_VALID_MSGID CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
- #define CFE_SB_DEFAULT_ROUTING_FILENAME CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME
- #define CFE_SB_DEFAULT_PIPE_FILENAME CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
- #define CFE_SB_DEFAULT_MAP_FILENAME CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
- #define CFE_SB_FILTERED_EVENT1 CFE_PLATFORM_SB_FILTERED_EVENT1
- #define CFE_SB_FILTER_MASK1 CFE_PLATFORM_SB_FILTER_MASK1
- #define CFE_SB_FILTERED_EVENT2 CFE_PLATFORM_SB_FILTERED_EVENT2
- #define CFE_SB_FILTER_MASK2 CFE_PLATFORM_SB_FILTER_MASK2
- #define CFE_SB_FILTERED_EVENT3 CFE_PLATFORM_SB_FILTERED_EVENT3
- #define CFE_SB_FILTER_MASK3 CFE_PLATFORM_SB_FILTER_MASK3
- #define CFE_SB_FILTERED_EVENT4 CFE_PLATFORM_SB_FILTERED_EVENT4
- #define CFE_SB_FILTER_MASK4 CFE_PLATFORM_SB_FILTER_MASK4
- #define CFE_SB_FILTERED_EVENT5 CFE_PLATFORM_SB_FILTERED_EVENT5
- #define CFE_SB_FILTER_MASK5 CFE_PLATFORM_SB_FILTER_MASK5
- #define CFE_SB_FILTERED_EVENT6 CFE_PLATFORM_SB_FILTERED_EVENT6
- #define CFE_SB_FILTER_MASK6 CFE_PLATFORM_SB_FILTER_MASK6
- #define CFE_SB_FILTERED_EVENT7 CFE_PLATFORM_SB_FILTERED_EVENT7
- #define CFE_SB_FILTER_MASK7 CFE_PLATFORM_SB_FILTER_MASK7
- #define CFE_SB_FILTERED_EVENT8 CFE_PLATFORM_SB_FILTERED_EVENT8
- #define CFE_SB_FILTER_MASK8 CFE_PLATFORM_SB_FILTER_MASK8
- #define CFE_SB_MEM_BLOCK_SIZE_01 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
- #define CFE_SB_MEM_BLOCK_SIZE_02 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02
- #define CFE_SB_MEM_BLOCK_SIZE_03 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03
- #define CFE_SB_MEM_BLOCK_SIZE_04 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04
- #define CFE_SB_MEM_BLOCK_SIZE_05 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05
- #define CFE_SB_MEM_BLOCK_SIZE_06 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06
- #define CFE_SB_MEM_BLOCK_SIZE_07 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07
- #define CFE_SB_MEM_BLOCK_SIZE_08 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08
- #define CFE_SB_MEM_BLOCK_SIZE_09 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09
- #define CFE_SB_MEM_BLOCK_SIZE_10 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10
- #define CFE_SB_MEM_BLOCK_SIZE_11 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11
- #define CFE_SB_MEM_BLOCK_SIZE_12 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12
- #define CFE_SB_MEM_BLOCK_SIZE_13 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13
- #define CFE_SB_MEM_BLOCK_SIZE_14 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14

- #define CFE_SB_MEM_BLOCK_SIZE_15 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15
- #define CFE_SB_MEM_BLOCK_SIZE_16 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16
- #define CFE_SB_MAX_BLOCK_SIZE CFE_PLATFORM_SB_MAX_BLOCK_SIZE
- #define CFE_SB_DEFAULT_REPORT_SENDER CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER
- #define CFE_TIME_CFG_SERVER CFE_PLATFORM_TIME_CFG_SERVER
- #define CFE_TIME_CFG_CLIENT CFE_PLATFORM_TIME_CFG_CLIENT
- #define CFE_TIME_CFG_VIRTUAL CFE_PLATFORM_TIME_CFG_VIRTUAL
- #define CFE_TIME_CFG_SIGNAL CFE_PLATFORM_TIME_CFG_SIGNAL
- #define CFE_TIME_CFG_SOURCE CFE_PLATFORM_TIME_CFG_SOURCE
- #define CFE_TIME_CFG_SRC_MET CFE_PLATFORM_TIME_CFG_SRC_MET
- #define CFE_TIME_CFG_SRC_GPS CFE_PLATFORM_TIME_CFG_SRC_GPS
- #define CFE_TIME_CFG_SRC_TIME CFE_PLATFORM_TIME_CFG_SRC_TIME
- #define CFE_TIME_MAX_DELTA_SECS CFE_PLATFORM_TIME_MAX_DELTA_SECS
- #define CFE_TIME_MAX_DELTA_SUBS CFE_PLATFORM_TIME_MAX_DELTA_SUBS
- #define CFE_TIME_MAX_LOCAL_SECS CFE_PLATFORM_TIME_MAX_LOCAL_SECS
- #define CFE_TIME_MAX_LOCAL_SUBS CFE_PLATFORM_TIME_MAX_LOCAL_SUBS
- #define CFE_TIME_CFG_TONE_LIMIT CFE_PLATFORM_TIME_CFG_TONE_LIMIT
- #define CFE_TIME_CFG_START_FLY CFE_PLATFORM_TIME_CFG_START_FLY
- #define CFE_TIME_CFG_LATCH_FLY CFE_PLATFORM_TIME_CFG_LATCH_FLY
- #define CFE_ES_MAX_APPLICATIONS CFE_PLATFORM_ES_MAX_APPLICATIONS
- #define CFE_ES_MAX_LIBRARIES CFE_PLATFORM_ES_MAX_LIBRARIES
- #define CFE_ES_ER_LOG_ENTRIES CFE_PLATFORM_ES_ER_LOG_ENTRIES
- #define CFE_ES_ER_LOG_MAX_CONTEXT_SIZE CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE
- #define CFE_ES_SYSTEM_LOG_SIZE CFE_PLATFORM_ES_SYSTEM_LOG_SIZE
- #define CFE_ES_OBJECT_TABLE_SIZE CFE_PLATFORM_ES_OBJECT_TABLE_SIZE
- #define CFE_ES_MAX_GEN_COUNTERS CFE_PLATFORM_ES_MAX_GEN_COUNTERS
- #define CFE_ES_APP_SCAN_RATE CFE_PLATFORM_ES_APP_SCAN_RATE
- #define CFE_ES_APP_KILL_TIMEOUT CFE_PLATFORM_ES_APP_KILL_TIMEOUT
- #define CFE_ES_RAM_DISK_SECTOR_SIZE CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE
- #define CFE_ES_RAM_DISK_NUM_SECTORS CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS
- #define CFE_ES_RAM_DISK_PERCENT_RESERVED CFE_PLATFORM_ES_RAM_DISK_PERCENT_RES↩ ERVED
- #define CFE_ES_RAM_DISK_MOUNT_STRING CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING
- #define CFE_ES_CDS_SIZE CFE_PLATFORM_ES_CDS_SIZE
- #define CFE_ES_USER_RESERVED_SIZE CFE_PLATFORM_ES_USER_RESERVED_SIZE
- #define CFE_ES_RESET_AREA_SIZE CFE_PLATFORM_ES_RESET_AREA_SIZE
- #define CFE_ES_NONVOL_STARTUP_FILE CFE_PLATFORM_ES_NONVOL_STARTUP_FILE
- #define CFE_ES_VOLATILE_STARTUP_FILE CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE
- #define CFE_ES_DEFAULT_SHELL_FILENAME CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME
- #define CFE_ES_MAX_SHELL_CMD CFE_PLATFORM_ES_MAX_SHELL_CMD
- #define CFE_ES_MAX_SHELL_PKT CFE_PLATFORM_ES_MAX_SHELL_PKT
- #define CFE_ES_DEFAULT_APP_LOG_FILE CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE
- #define CFE_ES_DEFAULT_TASK_LOG_FILE CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE
- #define CFE_ES_DEFAULT_SYSLOG_FILE CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE
- #define CFE_ES_DEFAULT_ER_LOG_FILE CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE
- #define CFE_ES_DEFAULT_PERF_DUMP_FILENAME CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FI↩ LENAME
- #define CFE_ES_DEFAULT_CDS_REG_DUMP_FILE CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP↩ _FILE
- #define CFE_ES_DEFAULT_SYSLOG_MODE CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE
- #define CFE_ES_PERF_MAX_IDS CFE_PLATFORM_ES_PERF_MAX_IDS

- #define CFE_ES_PERF_DATA_BUFFER_SIZE CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE
- #define CFE_ES_PERF_FILTMASK_NONE CFE_PLATFORM_ES_PERF_FILTMASK_NONE
- #define CFE_ES_PERF_FILTMASK_ALL CFE_PLATFORM_ES_PERF_FILTMASK_ALL
- #define CFE_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_PERF_FILTMASK_INIT
- #define CFE_ES_PERF_TRIGMASK_NONE CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
- #define CFE_ES_PERF_TRIGMASK_ALL CFE_PLATFORM_ES_PERF_TRIGMASK_ALL
- #define CFE_ES_PERF_TRIGMASK_INIT CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
- #define CFE_ES_PERF_CHILD_PRIORITY CFE_PLATFORM_ES_PERF_CHILD_PRIORITY
- #define CFE_ES_PERF_CHILD_STACK_SIZE CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE
- #define CFE_ES_PERF_CHILD_MS_DELAY CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY
- #define CFE_ES_PERF_ENTRIES_BTWN_DLYS CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS
- #define CFE_ES_DEFAULT_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- #define CFE_ES_EXCEPTION_FUNCTION CFE_PLATFORM_ES_EXCEPTION_FUNCTION
- #define CFE_EVS_START_TASK_PRIORITY CFE_PLATFORM_EVS_START_TASK_PRIORITY
- #define CFE_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
- #define CFE_SB_START_TASK_PRIORITY CFE_PLATFORM_SB_START_TASK_PRIORITY
- #define CFE_SB_START_TASK_STACK_SIZE CFE_PLATFORM_SB_START_TASK_STACK_SIZE
- #define CFE_ES_START_TASK_PRIORITY CFE_PLATFORM_ES_START_TASK_PRIORITY
- #define CFE_ES_START_TASK_STACK_SIZE CFE_PLATFORM_ES_START_TASK_STACK_SIZE
- #define CFE_TIME_START_TASK_PRIORITY CFE_PLATFORM_TIME_START_TASK_PRIORITY
- #define CFE_TIME_TONE_TASK_PRIORITY CFE_PLATFORM_TIME_TONE_TASK_PRIORITY
- #define CFE_TIME_1HZ_TASK_PRIORITY CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY
- #define CFE_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_TIME_START_TASK_STACK_SIZE
- #define CFE_TIME_TONE_TASK_STACK_SIZE CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE
- #define CFE_TIME_1HZ_TASK_STACK_SIZE CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE
- #define CFE_TBL_START_TASK_PRIORITY CFE_PLATFORM_TBL_START_TASK_PRIORITY
- #define CFE_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_TBL_START_TASK_STACK_SIZE
- #define CFE_ES_CDS_MAX_NUM_ENTRIES CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES
- #define CFE_ES_MAX_PROCESSOR_RESETS CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS
- #define CFE_ES_MEM_BLOCK_SIZE_01 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01
- #define CFE_ES_MEM_BLOCK_SIZE_02 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02
- #define CFE_ES_MEM_BLOCK_SIZE_03 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03
- #define CFE_ES_MEM_BLOCK_SIZE_04 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04
- #define CFE_ES_MEM_BLOCK_SIZE_05 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05
- #define CFE_ES_MEM_BLOCK_SIZE_06 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06
- #define CFE_ES_MEM_BLOCK_SIZE_07 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07
- #define CFE_ES_MEM_BLOCK_SIZE_08 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08
- #define CFE_ES_MEM_BLOCK_SIZE_09 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09
- #define CFE_ES_MEM_BLOCK_SIZE_10 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10
- #define CFE_ES_MEM_BLOCK_SIZE_11 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11
- #define CFE_ES_MEM_BLOCK_SIZE_12 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12
- #define CFE_ES_MEM_BLOCK_SIZE_13 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13
- #define CFE_ES_MEM_BLOCK_SIZE_14 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14
- #define CFE_ES_MEM_BLOCK_SIZE_15 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15
- #define CFE_ES_MEM_BLOCK_SIZE_16 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16
- #define CFE_ES_MAX_BLOCK_SIZE CFE_PLATFORM_ES_MAX_BLOCK_SIZE
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_01 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_02 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_03 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_04 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_05 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05

- #define CFE_ES_CDS_MEM_BLOCK_SIZE_06 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_07 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_08 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_09 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_10 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_11 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_12 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_13 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_14 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_15 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15
- #define CFE_ES_CDS_MEM_BLOCK_SIZE_16 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16
- #define CFE_ES_CDS_MAX_BLOCK_SIZE CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE
- #define CFE_EVS_MAX_EVENT_FILTERS CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
- #define CFE_EVS_LOG_ON CFE_PLATFORM_EVS_LOG_ON
- #define CFE_EVS_DEFAULT_LOG_FILE CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
- #define CFE_EVS_LOG_MAX CFE_PLATFORM_EVS_LOG_MAX
- #define CFE_EVS_DEFAULT_APP_DATA_FILE CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE
- #define CFE_EVS_PORT_DEFAULT CFE_PLATFORM_EVS_PORT_DEFAULT
- #define CFE_EVS_DEFAULT_TYPE_FLAG CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG
- #define CFE_EVS_DEFAULT_LOG_MODE CFE_PLATFORM_EVS_DEFAULT_LOG_MODE
- #define CFE_EVS_DEFAULT_MSG_FORMAT_MODE CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_↩ MODE
- #define CFE_TBL_BUF_MEMORY_BYTES CFE_PLATFORM_TBL_BUF_MEMORY_BYTES
- #define CFE_TBL_MAX_DBL_TABLE_SIZE CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE
- #define CFE_TBL_MAX_SNGL_TABLE_SIZE CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE
- #define CFE_TBL_MAX_NUM_TABLES CFE_PLATFORM_TBL_MAX_NUM_TABLES
- #define CFE_TBL_MAX_CRITICAL_TABLES CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES
- #define CFE_TBL_MAX_NUM_HANDLES CFE_PLATFORM_TBL_MAX_NUM_HANDLES
- #define CFE_TBL_MAX_SIMULTANEOUS_LOADS CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS
- #define CFE_TBL_MAX_NUM_VALIDATIONS CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS
- #define CFE_TBL_DEFAULT_REG_DUMP_FILE CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE
- #define CFE_TBL_VALID_SCID_COUNT CFE_PLATFORM_TBL_VALID_SCID_COUNT
- #define CFE_TBL_U32FROM4CHARS CFE_PLATFORM_TBL_U32FROM4CHARS
- #define CFE_TBL_VALID_SCID_1 CFE_PLATFORM_TBL_VALID_SCID_1
- #define CFE_TBL_VALID_SCID_2 CFE_PLATFORM_TBL_VALID_SCID_2
- #define CFE_TBL_VALID_PRID_COUNT CFE_PLATFORM_TBL_VALID_PRID_COUNT
- #define CFE_TBL_VALID_PRID_1 CFE_PLATFORM_TBL_VALID_PRID_1
- #define CFE_TBL_VALID_PRID_2 CFE_PLATFORM_TBL_VALID_PRID_2
- #define CFE_TBL_VALID_PRID_3 CFE_PLATFORM_TBL_VALID_PRID_3
- #define CFE_TBL_VALID_PRID_4 CFE_PLATFORM_TBL_VALID_PRID_4
- #define CFE_ES_STARTUP_SYNC_POLL_MSEC CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC
- #define CFE_CORE_MAX_STARTUP_MSEC CFE_PLATFORM_CORE_MAX_STARTUP_MSEC
- #define CFE_ES_STARTUP_SCRIPT_TIMEOUT_MSEC CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEO↩ UT_MSEC
- #define CFE_TIME_ENA_1HZ_CMD_PKT true

13.2.1 Macro Definition Documentation

13.2.1.1 CFE_CORE_MAX_STARTUP_MSEC

```
#define CFE_CORE_MAX_STARTUP_MSEC CFE_PLATFORM_CORE_MAX_STARTUP_MSEC
```

Definition at line 2093 of file cpu1_platform_cfg.h.

13.2.1.2 CFE_CPU_ID

```
#define CFE_CPU_ID CFE_PLATFORM_CPU_ID
```

Definition at line 1912 of file cpu1_platform_cfg.h.

13.2.1.3 CFE_CPU_NAME

```
#define CFE_CPU_NAME CFE_PLATFORM_CPU_NAME
```

Definition at line 1913 of file cpu1_platform_cfg.h.

13.2.1.4 CFE_ES_APP_KILL_TIMEOUT

```
#define CFE_ES_APP_KILL_TIMEOUT CFE_PLATFORM_ES_APP_KILL_TIMEOUT
```

Definition at line 1981 of file cpu1_platform_cfg.h.

13.2.1.5 CFE_ES_APP_SCAN_RATE

```
#define CFE_ES_APP_SCAN_RATE CFE_PLATFORM_ES_APP_SCAN_RATE
```

Definition at line 1980 of file cpu1_platform_cfg.h.

13.2.1.6 CFE_ES_CDS_MAX_BLOCK_SIZE

```
#define CFE_ES_CDS_MAX_BLOCK_SIZE CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE
```

Definition at line 2064 of file cpu1_platform_cfg.h.

13.2.1.7 CFE_ES_CDS_MAX_NUM_ENTRIES

```
#define CFE_ES_CDS_MAX_NUM_ENTRIES CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES
```

Definition at line 2029 of file cpu1_platform_cfg.h.

13.2.1.8 CFE_ES_CDS_MEM_BLOCK_SIZE_01

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_01 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
```

Definition at line 2048 of file cpu1_platform_cfg.h.

13.2.1.9 CFE_ES_CDS_MEM_BLOCK_SIZE_02

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_02 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02
```

Definition at line 2049 of file cpu1_platform_cfg.h.

13.2.1.10 CFE_ES_CDS_MEM_BLOCK_SIZE_03

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_03 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03
```

Definition at line 2050 of file cpu1_platform_cfg.h.

13.2.1.11 CFE_ES_CDS_MEM_BLOCK_SIZE_04

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_04 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04
```

Definition at line 2051 of file cpu1_platform_cfg.h.

13.2.1.12 CFE_ES_CDS_MEM_BLOCK_SIZE_05

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_05 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05
```

Definition at line 2052 of file cpu1_platform_cfg.h.

13.2.1.13 CFE_ES_CDS_MEM_BLOCK_SIZE_06

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_06 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06
```

Definition at line 2053 of file cpu1_platform_cfg.h.

13.2.1.14 CFE_ES_CDS_MEM_BLOCK_SIZE_07

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_07 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07
```

Definition at line 2054 of file cpu1_platform_cfg.h.

13.2.1.15 CFE_ES_CDS_MEM_BLOCK_SIZE_08

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_08 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08
```

Definition at line 2055 of file cpu1_platform_cfg.h.

13.2.1.16 CFE_ES_CDS_MEM_BLOCK_SIZE_09

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_09 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09
```

Definition at line 2056 of file cpu1_platform_cfg.h.

13.2.1.17 CFE_ES_CDS_MEM_BLOCK_SIZE_10

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_10 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10
```

Definition at line 2057 of file cpu1_platform_cfg.h.

13.2.1.18 CFE_ES_CDS_MEM_BLOCK_SIZE_11

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_11 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11
```

Definition at line 2058 of file cpu1_platform_cfg.h.

13.2.1.19 CFE_ES_CDS_MEM_BLOCK_SIZE_12

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_12 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12
```

Definition at line 2059 of file cpu1_platform_cfg.h.

13.2.1.20 CFE_ES_CDS_MEM_BLOCK_SIZE_13

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_13 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13
```

Definition at line 2060 of file cpu1_platform_cfg.h.

13.2.1.21 CFE_ES_CDS_MEM_BLOCK_SIZE_14

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_14 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14
```

Definition at line 2061 of file cpu1_platform_cfg.h.

13.2.1.22 CFE_ES_CDS_MEM_BLOCK_SIZE_15

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_15 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15
```

Definition at line 2062 of file cpu1_platform_cfg.h.

13.2.1.23 CFE_ES_CDS_MEM_BLOCK_SIZE_16

```
#define CFE_ES_CDS_MEM_BLOCK_SIZE_16 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16
```

Definition at line 2063 of file cpu1_platform_cfg.h.

13.2.1.24 CFE_ES_CDS_SIZE

```
#define CFE_ES_CDS_SIZE CFE_PLATFORM_ES_CDS_SIZE
```

Definition at line 1986 of file cpu1_platform_cfg.h.

13.2.1.25 CFE_ES_DEFAULT_APP_LOG_FILE

```
#define CFE_ES_DEFAULT_APP_LOG_FILE CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE
```

Definition at line 1994 of file cpu1_platform_cfg.h.

13.2.1.26 CFE_ES_DEFAULT_CDS_REG_DUMP_FILE

```
#define CFE_ES_DEFAULT_CDS_REG_DUMP_FILE CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE
```

Definition at line 1999 of file cpu1_platform_cfg.h.

13.2.1.27 CFE_ES_DEFAULT_ER_LOG_FILE

```
#define CFE_ES_DEFAULT_ER_LOG_FILE CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE
```

Definition at line 1997 of file cpu1_platform_cfg.h.

13.2.1.28 CFE_ES_DEFAULT_PERF_DUMP_FILENAME

```
#define CFE_ES_DEFAULT_PERF_DUMP_FILENAME CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME
```

Definition at line 1998 of file cpu1_platform_cfg.h.

13.2.1.29 CFE_ES_DEFAULT_SHELL_FILENAME

```
#define CFE_ES_DEFAULT_SHELL_FILENAME CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME
```

Definition at line 1991 of file cpu1_platform_cfg.h.

13.2.1.30 CFE_ES_DEFAULT_STACK_SIZE

```
#define CFE_ES_DEFAULT_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Definition at line 2013 of file cpu1_platform_cfg.h.

13.2.1.31 CFE_ES_DEFAULT_SYSLOG_FILE

```
#define CFE_ES_DEFAULT_SYSLOG_FILE CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE
```

Definition at line 1996 of file cpu1_platform_cfg.h.

13.2.1.32 CFE_ES_DEFAULT_SYSLOG_MODE

```
#define CFE_ES_DEFAULT_SYSLOG_MODE CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE
```

Definition at line 2000 of file cpu1_platform_cfg.h.

13.2.1.33 CFE_ES_DEFAULT_TASK_LOG_FILE

```
#define CFE_ES_DEFAULT_TASK_LOG_FILE CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE
```

Definition at line 1995 of file cpu1_platform_cfg.h.

13.2.1.34 CFE_ES_ER_LOG_ENTRIES

```
#define CFE_ES_ER_LOG_ENTRIES CFE_PLATFORM_ES_ER_LOG_ENTRIES
```

Definition at line 1975 of file cpu1_platform_cfg.h.

13.2.1.35 CFE_ES_ER_LOG_MAX_CONTEXT_SIZE

```
#define CFE_ES_ER_LOG_MAX_CONTEXT_SIZE CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE
```

Definition at line 1976 of file cpu1_platform_cfg.h.

13.2.1.36 CFE_ES_EXCEPTION_FUNCTION

```
#define CFE_ES_EXCEPTION_FUNCTION CFE_PLATFORM_ES_EXCEPTION_FUNCTION
```

Definition at line 2014 of file cpu1_platform_cfg.h.

13.2.1.37 CFE_ES_MAX_APPLICATIONS

```
#define CFE_ES_MAX_APPLICATIONS CFE_PLATFORM_ES_MAX_APPLICATIONS
```

Definition at line 1973 of file cpu1_platform_cfg.h.

13.2.1.38 CFE_ES_MAX_BLOCK_SIZE

```
#define CFE_ES_MAX_BLOCK_SIZE CFE_PLATFORM_ES_MAX_BLOCK_SIZE
```

Definition at line 2047 of file cpu1_platform_cfg.h.

13.2.1.39 CFE_ES_MAX_GEN_COUNTERS

```
#define CFE_ES_MAX_GEN_COUNTERS CFE_PLATFORM_ES_MAX_GEN_COUNTERS
```

Definition at line 1979 of file cpu1_platform_cfg.h.

13.2.1.40 CFE_ES_MAX_LIBRARIES

```
#define CFE_ES_MAX_LIBRARIES CFE_PLATFORM_ES_MAX_LIBRARIES
```

Definition at line 1974 of file cpu1_platform_cfg.h.

13.2.1.41 CFE_ES_MAX_PROCESSOR_RESETS

```
#define CFE_ES_MAX_PROCESSOR_RESETS CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS
```

Definition at line 2030 of file cpu1_platform_cfg.h.

13.2.1.42 CFE_ES_MAX_SHELL_CMD

```
#define CFE_ES_MAX_SHELL_CMD CFE_PLATFORM_ES_MAX_SHELL_CMD
```

Definition at line 1992 of file cpu1_platform_cfg.h.

13.2.1.43 CFE_ES_MAX_SHELL_PKT

```
#define CFE_ES_MAX_SHELL_PKT CFE_PLATFORM_ES_MAX_SHELL_PKT
```

Definition at line 1993 of file cpu1_platform_cfg.h.

13.2.1.44 CFE_ES_MEM_BLOCK_SIZE_01

```
#define CFE_ES_MEM_BLOCK_SIZE_01 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01
```

Definition at line 2031 of file cpu1_platform_cfg.h.

13.2.1.45 CFE_ES_MEM_BLOCK_SIZE_02

```
#define CFE_ES_MEM_BLOCK_SIZE_02 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02
```

Definition at line 2032 of file cpu1_platform_cfg.h.

13.2.1.46 CFE_ES_MEM_BLOCK_SIZE_03

```
#define CFE_ES_MEM_BLOCK_SIZE_03 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03
```

Definition at line 2033 of file cpu1_platform_cfg.h.

13.2.1.47 CFE_ES_MEM_BLOCK_SIZE_04

```
#define CFE_ES_MEM_BLOCK_SIZE_04 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04
```

Definition at line 2034 of file cpu1_platform_cfg.h.

13.2.1.48 CFE_ES_MEM_BLOCK_SIZE_05

```
#define CFE_ES_MEM_BLOCK_SIZE_05 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05
```

Definition at line 2035 of file cpu1_platform_cfg.h.

13.2.1.49 CFE_ES_MEM_BLOCK_SIZE_06

```
#define CFE_ES_MEM_BLOCK_SIZE_06 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06
```

Definition at line 2036 of file cpu1_platform_cfg.h.

13.2.1.50 CFE_ES_MEM_BLOCK_SIZE_07

```
#define CFE_ES_MEM_BLOCK_SIZE_07 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07
```

Definition at line 2037 of file cpu1_platform_cfg.h.

13.2.1.51 CFE_ES_MEM_BLOCK_SIZE_08

```
#define CFE_ES_MEM_BLOCK_SIZE_08 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08
```

Definition at line 2038 of file cpu1_platform_cfg.h.

13.2.1.52 CFE_ES_MEM_BLOCK_SIZE_09

```
#define CFE_ES_MEM_BLOCK_SIZE_09 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09
```

Definition at line 2039 of file cpu1_platform_cfg.h.

13.2.1.53 CFE_ES_MEM_BLOCK_SIZE_10

```
#define CFE_ES_MEM_BLOCK_SIZE_10 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10
```

Definition at line 2040 of file cpu1_platform_cfg.h.

13.2.1.54 CFE_ES_MEM_BLOCK_SIZE_11

```
#define CFE_ES_MEM_BLOCK_SIZE_11 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11
```

Definition at line 2041 of file cpu1_platform_cfg.h.

13.2.1.55 CFE_ES_MEM_BLOCK_SIZE_12

```
#define CFE_ES_MEM_BLOCK_SIZE_12 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12
```

Definition at line 2042 of file cpu1_platform_cfg.h.

13.2.1.56 CFE_ES_MEM_BLOCK_SIZE_13

```
#define CFE_ES_MEM_BLOCK_SIZE_13 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13
```

Definition at line 2043 of file cpu1_platform_cfg.h.

13.2.1.57 CFE_ES_MEM_BLOCK_SIZE_14

```
#define CFE_ES_MEM_BLOCK_SIZE_14 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14
```

Definition at line 2044 of file cpu1_platform_cfg.h.

13.2.1.58 CFE_ES_MEM_BLOCK_SIZE_15

```
#define CFE_ES_MEM_BLOCK_SIZE_15 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15
```

Definition at line 2045 of file cpu1_platform_cfg.h.

13.2.1.59 CFE_ES_MEM_BLOCK_SIZE_16

```
#define CFE_ES_MEM_BLOCK_SIZE_16 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16
```

Definition at line 2046 of file cpu1_platform_cfg.h.

13.2.1.60 CFE_ES_NONVOL_STARTUP_FILE

```
#define CFE_ES_NONVOL_STARTUP_FILE CFE_PLATFORM_ES_NONVOL_STARTUP_FILE
```

Definition at line 1989 of file cpu1_platform_cfg.h.

13.2.1.61 CFE_ES_OBJECT_TABLE_SIZE

```
#define CFE_ES_OBJECT_TABLE_SIZE CFE_PLATFORM_ES_OBJECT_TABLE_SIZE
```

Definition at line 1978 of file cpu1_platform_cfg.h.

13.2.1.62 CFE_ES_PERF_CHILD_MS_DELAY

```
#define CFE_ES_PERF_CHILD_MS_DELAY CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY
```

Definition at line 2011 of file cpu1_platform_cfg.h.

13.2.1.63 CFE_ES_PERF_CHILD_PRIORITY

```
#define CFE_ES_PERF_CHILD_PRIORITY CFE_PLATFORM_ES_PERF_CHILD_PRIORITY
```

Definition at line 2009 of file cpu1_platform_cfg.h.

13.2.1.64 CFE_ES_PERF_CHILD_STACK_SIZE

```
#define CFE_ES_PERF_CHILD_STACK_SIZE CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE
```

Definition at line 2010 of file cpu1_platform_cfg.h.

13.2.1.65 CFE_ES_PERF_DATA_BUFFER_SIZE

```
#define CFE_ES_PERF_DATA_BUFFER_SIZE CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE
```

Definition at line 2002 of file cpu1_platform_cfg.h.

13.2.1.66 CFE_ES_PERF_ENTRIES_BTWN_DLYS

```
#define CFE_ES_PERF_ENTRIES_BTWN_DLYS CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS
```

Definition at line 2012 of file cpu1_platform_cfg.h.

13.2.1.67 CFE_ES_PERF_FILTMASK_ALL

```
#define CFE_ES_PERF_FILTMASK_ALL CFE_PLATFORM_ES_PERF_FILTMASK_ALL
```

Definition at line 2004 of file cpu1_platform_cfg.h.

13.2.1.68 CFE_ES_PERF_FILTMASK_INIT

```
#define CFE_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_PERF_FILTMASK_INIT
```

Definition at line 2005 of file cpu1_platform_cfg.h.

13.2.1.69 CFE_ES_PERF_FILTMASK_NONE

```
#define CFE_ES_PERF_FILTMASK_NONE CFE_PLATFORM_ES_PERF_FILTMASK_NONE
```

Definition at line 2003 of file cpu1_platform_cfg.h.

13.2.1.70 CFE_ES_PERF_MAX_IDS

```
#define CFE_ES_PERF_MAX_IDS CFE_PLATFORM_ES_PERF_MAX_IDS
```

Definition at line 2001 of file cpu1_platform_cfg.h.

13.2.1.71 CFE_ES_PERF_TRIGMASK_ALL

```
#define CFE_ES_PERF_TRIGMASK_ALL CFE_PLATFORM_ES_PERF_TRIGMASK_ALL
```

Definition at line 2007 of file cpu1_platform_cfg.h.

13.2.1.72 CFE_ES_PERF_TRIGMASK_INIT

```
#define CFE_ES_PERF_TRIGMASK_INIT CFE_PLATFORM_ES_PERF_TRIGMASK_INIT
```

Definition at line 2008 of file cpu1_platform_cfg.h.

13.2.1.73 CFE_ES_PERF_TRIGMASK_NONE

```
#define CFE_ES_PERF_TRIGMASK_NONE CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

Definition at line 2006 of file cpu1_platform_cfg.h.

13.2.1.74 CFE_ES_RAM_DISK_MOUNT_STRING

```
#define CFE_ES_RAM_DISK_MOUNT_STRING CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING
```

Definition at line 1985 of file cpu1_platform_cfg.h.

13.2.1.75 CFE_ES_RAM_DISK_NUM_SECTORS

```
#define CFE_ES_RAM_DISK_NUM_SECTORS CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS
```

Definition at line 1983 of file cpu1_platform_cfg.h.

13.2.1.76 CFE_ES_RAM_DISK_PERCENT_RESERVED

```
#define CFE_ES_RAM_DISK_PERCENT_RESERVED CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED
```

Definition at line 1984 of file cpu1_platform_cfg.h.

13.2.1.77 CFE_ES_RAM_DISK_SECTOR_SIZE

```
#define CFE_ES_RAM_DISK_SECTOR_SIZE CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE
```

Definition at line 1982 of file cpu1_platform_cfg.h.

13.2.1.78 CFE_ES_RESET_AREA_SIZE

```
#define CFE_ES_RESET_AREA_SIZE CFE_PLATFORM_ES_RESET_AREA_SIZE
```

Definition at line 1988 of file cpu1_platform_cfg.h.

13.2.1.79 CFE_ES_START_TASK_PRIORITY

```
#define CFE_ES_START_TASK_PRIORITY CFE_PLATFORM_ES_START_TASK_PRIORITY
```

Definition at line 2019 of file cpu1_platform_cfg.h.

13.2.1.80 CFE_ES_START_TASK_STACK_SIZE

```
#define CFE_ES_START_TASK_STACK_SIZE CFE_PLATFORM_ES_START_TASK_STACK_SIZE
```

Definition at line 2020 of file cpu1_platform_cfg.h.

13.2.1.81 CFE_ES_STARTUP_SCRIPT_TIMEOUT_MSEC

```
#define CFE_ES_STARTUP_SCRIPT_TIMEOUT_MSEC CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC
```

Definition at line 2094 of file cpu1_platform_cfg.h.

13.2.1.82 CFE_ES_STARTUP_SYNC_POLL_MSEC

```
#define CFE_ES_STARTUP_SYNC_POLL_MSEC CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC
```

Definition at line 2092 of file cpu1_platform_cfg.h.

13.2.1.83 CFE_ES_SYSTEM_LOG_SIZE

```
#define CFE_ES_SYSTEM_LOG_SIZE CFE_PLATFORM_ES_SYSTEM_LOG_SIZE
```

Definition at line 1977 of file cpu1_platform_cfg.h.

13.2.1.84 CFE_ES_USER_RESERVED_SIZE

```
#define CFE_ES_USER_RESERVED_SIZE CFE_PLATFORM_ES_USER_RESERVED_SIZE
```

Definition at line 1987 of file cpu1_platform_cfg.h.

13.2.1.85 CFE_ES_VOLATILE_STARTUP_FILE

```
#define CFE_ES_VOLATILE_STARTUP_FILE CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE
```

Definition at line 1990 of file cpu1_platform_cfg.h.

13.2.1.86 CFE_EVS_DEFAULT_APP_DATA_FILE

```
#define CFE_EVS_DEFAULT_APP_DATA_FILE CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE
```

Definition at line 2069 of file cpu1_platform_cfg.h.

13.2.1.87 CFE_EVS_DEFAULT_LOG_FILE

```
#define CFE_EVS_DEFAULT_LOG_FILE CFE_PLATFORM_EVS_DEFAULT_LOG_FILE
```

Definition at line 2067 of file cpu1_platform_cfg.h.

13.2.1.88 CFE_EVS_DEFAULT_LOG_MODE

```
#define CFE_EVS_DEFAULT_LOG_MODE CFE_PLATFORM_EVS_DEFAULT_LOG_MODE
```

Definition at line 2072 of file cpu1_platform_cfg.h.

13.2.1.89 CFE_EVS_DEFAULT_MSG_FORMAT_MODE

```
#define CFE_EVS_DEFAULT_MSG_FORMAT_MODE CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE
```

Definition at line 2073 of file cpu1_platform_cfg.h.

13.2.1.90 CFE_EVS_DEFAULT_TYPE_FLAG

```
#define CFE_EVS_DEFAULT_TYPE_FLAG CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG
```

Definition at line 2071 of file cpu1_platform_cfg.h.

13.2.1.91 CFE_EVS_LOG_MAX

```
#define CFE_EVS_LOG_MAX CFE_PLATFORM_EVS_LOG_MAX
```

Definition at line 2068 of file cpu1_platform_cfg.h.

13.2.1.92 CFE_EVS_LOG_ON

```
#define CFE_EVS_LOG_ON CFE_PLATFORM_EVS_LOG_ON
```

Definition at line 2066 of file cpu1_platform_cfg.h.

13.2.1.93 CFE_EVS_MAX_EVENT_FILTERS

```
#define CFE_EVS_MAX_EVENT_FILTERS CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
```

Definition at line 2065 of file cpu1_platform_cfg.h.

13.2.1.94 CFE_EVS_PORT_DEFAULT

```
#define CFE_EVS_PORT_DEFAULT CFE_PLATFORM_EVS_PORT_DEFAULT
```

Definition at line 2070 of file cpu1_platform_cfg.h.

13.2.1.95 CFE_EVS_START_TASK_PRIORITY

```
#define CFE_EVS_START_TASK_PRIORITY CFE_PLATFORM_EVS_START_TASK_PRIORITY
```

Definition at line 2015 of file cpu1_platform_cfg.h.

13.2.1.96 CFE_EVS_START_TASK_STACK_SIZE

```
#define CFE_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
```

Definition at line 2016 of file cpu1_platform_cfg.h.

13.2.1.97 CFE_MISSION_REV

```
#define CFE_MISSION_REV 0
```

Purpose Mission specific version number for cFE

Description:

The cFE version number consists of four parts: major version number, minor version number, revision number and mission specific revision number. The mission specific revision number is defined here and the other parts are defined in "cfe_version.h".

Limits:

Must be defined as a numeric value that is greater than or equal to zero.

Definition at line 1830 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_NoopCmd(), CFE_ES_TaskInit(), CFE_EVS_NoopCmd(), CFE_EVS_TaskInit(), and CFE_↩ SB_NoopCmd().

13.2.1.98 CFE_PLATFORM_CORE_MAX_STARTUP_MSEC

```
#define CFE_PLATFORM_CORE_MAX_STARTUP_MSEC 30000
```

Purpose CFE core application startup timeout

Description:

The upper limit for the amount of time that the cFE core applications (ES, SB, EVS, TIME, TBL) are each alloted to reach their respective "ready" states.

The CFE "main" thread starts individual tasks for each of the core applications (except FS). Each of these must perform some initialization work before the next core application can be started, so the main thread waits to ensure that the application has reached the "ready" state before starting the next application.

If any core application fails to start, then it indicates a major problem with the system and startup is aborted.

Units are in milliseconds

Limits:

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 1876 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_CreateObjects(), CFE_ES_TaskMain(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

13.2.1.99 CFE_PLATFORM_CPU_ID

```
#define CFE_PLATFORM_CPU_ID 1
```

Definition at line 47 of file cpu1_platform_cfg.h.

13.2.1.100 CFE_PLATFORM_CPU_NAME

```
#define CFE_PLATFORM_CPU_NAME "CPU1"
```

Definition at line 52 of file cpu1_platform_cfg.h.

13.2.1.101 CFE_PLATFORM_ENDIAN

```
#define CFE_PLATFORM_ENDIAN CCSDS_LITTLE_ENDIAN
```

Purpose Platform Endian Indicator

Description:

The value of this constant indicates the endianess of the target system

Limits

This parameter has a lower limit of 0 and an upper limit of 1.

Definition at line 194 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_SetMsgId().

13.2.1.102 CFE_PLATFORM_ES_APP_KILL_TIMEOUT

```
#define CFE_PLATFORM_ES_APP_KILL_TIMEOUT 5
```

Purpose Define ES Application Kill Timeout

Description:

ES Application Kill Timeout. This parameter controls the number of "scan periods" that ES will wait for an application to Exit after getting the signal Delete, Reload or Restart. The sequence works as follows:

1. ES will set the control request for an App to Delete/Restart/Reload and set this kill timer to the value in this parameter.
2. If the App is reponding and Calls it's RunLoop function, it will drop out of it's main loop and call CFE_ES_↩ ExitApp. Once it calls Exit App, then ES can delete, restart, or reload the app the next time it scans the app table.
3. If the App is not responding, the ES App will decrement this Kill Timeout value each time it runs. If the timeout value reaches zero, ES will kill the app.

The Kill timeout value depends on the CFE_PLATFORM_ES_APP_SCAN_RATE. If the Scan Rate is 1000, or 1 second, and this CFE_PLATFORM_ES_APP_KILL_TIMEOUT is set to 5, then it will take 5 seconds to kill a non-responding App. If the Scan Rate is 250, or 1/4 second, and the CFE_PLATFORM_ES_APP_KILL_TIMEOUT is set to 2, then it will take 1/2 second to time out.

Limits

There is a lower limit of 1 and an upper limit of 100 on this configuration paramater. Units are number of CFE_P↩ LATFORM_ES_APP_SCAN_RATE cycles.

Definition at line 661 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_DeleteApp(), CFE_ES_ReloadApp(), and CFE_ES_RestartApp().

13.2.1.103 CFE_PLATFORM_ES_APP_SCAN_RATE

```
#define CFE_PLATFORM_ES_APP_SCAN_RATE 1000
```

Purpose Define ES Application Control Scan Rate

Description:

ES Application Control Scan Rate. This parameter controls the speed that ES scans the Application Table looking for App Delete/Restart/Reload requests. All Applications are deleted, restarted, or reloaded by the ES Application. ES will periodically scan for control requests to process. The scan rate is controlled by this parameter, which is given in milliseconds. A value of 1000 means that ES will scan the Application Table once per second. Be careful not to set the value of this too low, because ES will use more CPU cycles scanning the table.

Limits

There is a lower limit of 100 and an upper limit of 20000 on this configuration paramater. millisecond units.

Definition at line 631 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_TaskMain().

13.2.1.104 CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE 80000
```

Definition at line 1468 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_GetCDSBlock().

13.2.1.105 CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES

```
#define CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES 512
```

Purpose Define Maximum Number of Registered CDS Blocks

Description:

Maximum number of registered CDS Blocks

Limits

There is a lower limit of 8. There are no restrictions on the upper limit however, the maximum number of CDS entries is system dependent and should be verified.

Definition at line 1387 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_InitCDSRegistry(), and CFE_ES_RebuildCDS().

13.2.1.106 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01 8
```

Purpose Define ES Critical Data Store Memory Pool Block Sizes

Description:

Intermediate ES Critical Data Store Memory Pool Block Sizes

Limits

These sizes MUST be increasing and MUST be an integral multiple of 4.

Definition at line 1452 of file cpu1_platform_cfg.h.

13.2.1.107 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 16
```

Definition at line 1453 of file cpu1_platform_cfg.h.

13.2.1.108 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 32
```

Definition at line 1454 of file cpu1_platform_cfg.h.

13.2.1.109 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 48
```

Definition at line 1455 of file cpu1_platform_cfg.h.

13.2.1.110 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 64
```

Definition at line 1456 of file cpu1_platform_cfg.h.

13.2.1.111 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06 96
```

Definition at line 1457 of file cpu1_platform_cfg.h.

13.2.1.112 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07 128
```

Definition at line 1458 of file cpu1_platform_cfg.h.

13.2.1.113 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08 160
```

Definition at line 1459 of file cpu1_platform_cfg.h.

13.2.1.114 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09 256
```

Definition at line 1460 of file cpu1_platform_cfg.h.

13.2.1.115 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10 512
```

Definition at line 1461 of file cpu1_platform_cfg.h.

13.2.1.116 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11 1024
```

Definition at line 1462 of file cpu1_platform_cfg.h.

13.2.1.117 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12 2048
```

Definition at line 1463 of file cpu1_platform_cfg.h.

13.2.1.118 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13 4096
```

Definition at line 1464 of file cpu1_platform_cfg.h.

13.2.1.119 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14 8192
```

Definition at line 1465 of file cpu1_platform_cfg.h.

13.2.1.120 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15 16384
```

Definition at line 1466 of file cpu1_platform_cfg.h.

13.2.1.121 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16

```
#define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 32768
```

Definition at line 1467 of file cpu1_platform_cfg.h.

13.2.1.122 CFE_PLATFORM_ES_CDS_SIZE

```
#define CFE_PLATFORM_ES_CDS_SIZE ( 128 * 1024 )
```

Purpose Define Critical Data Store Size

Description:

Defines the Critical Data Store (CDS) area size in bytes size. The CDS is one of four memory areas that are preserved during a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 8192 and an upper limit of UINT_MAX (4 Gigabytes) on this configuration paramater.

Definition at line 758 of file cpu1_platform_cfg.h.

13.2.1.123 CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE

```
#define CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE "/ram/cfe_es_app_info.log"
```

Purpose Default Application Information Filename

Description:

The value of this constant defines the filename used to store information pertaining to all of the Applications that are registered with Executive Services. This filename is used only when no filename is specified in the the command to query all system apps.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 930 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_QueryAllCmd().

13.2.1.124 CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE

```
#define CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE "/ram/cfe_cds_reg.log"
```

Purpose Default Critical Data Store Registry Filename

Description:

The value of this constant defines the filename used to store the Critical Data Store Registry. This filename is used only when no filename is specified in the command to stop performance data collecting.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 1005 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

13.2.1.125 CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE

```
#define CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE "/ram/cfe_erlog.log"
```

Purpose Default Exception and Reset (ER) Log Filename

Description:

The value of this constant defines the filename used to store the Exception and Reset (ER) Log. This filename is used only when no filename is specified in the command to dump the ER log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 976 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_WriteERLogCmd().

13.2.1.126 CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME

```
#define CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME "/ram/cfe_es_perf.dat"
```

Purpose Default Performance Data Filename

Description:

The value of this constant defines the filename used to store the Performance Data. This filename is used only when no filename is specified in the command to stop performance data collecting.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 990 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.2.1.127 CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME

```
#define CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME "/ram/ShellCmd.out"
```

Purpose Default Shell Filename

Description:

The value of this constant defines the filename used to store the shell output after a shell command is received by ES. This file contains the entire shell output. The fsw also sends the shell output in series of fixed size telemetry packets. This filename is used only when no filename is specified in the shell command.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 868 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_ShellOutputCommand().

13.2.1.128 CFE_PLATFORM_ES_DEFAULT_STACK_SIZE

```
#define CFE_PLATFORM_ES_DEFAULT_STACK_SIZE 8192
```

Purpose Define Default Stack Size for an Application

Description:

This parameter defines a default stack size. This parameter is used by the cFE Core Applications.

Limits

There is a lower limit of 2048. There are no restrictions on the upper limit however, the maximum stack size size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1188 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_StartAppCmd().

13.2.1.129 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE

```
#define CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE "/ram/cfe_es_syslog.log"
```

Purpose Default System Log Filename

Description:

The value of this constant defines the filename used to store important information (as ASCII text strings) that might not be able to be sent in an Event Message. This filename is used only when no filename is specified in the command to dump the system log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 961 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_WriteSyslogCmd().

13.2.1.130 CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE

```
#define CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE 1
```

Purpose Define Default System Log Mode

Description:

Defines the default mode for the operation of the ES System log. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest message in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. This constant may hold a value of either 0 or 1 depending on the desired default log mode. Overwrite Mode = 0, Discard Mode = 1.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration paramater.

Definition at line 1023 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_TaskInit().

13.2.1.131 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE

```
#define CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE "/ram/cfe_es_task_info.log"
```

Purpose Default Application Information Filename

Description:

The value of this constant defines the filename used to store information pertaining to all of the Applications that are registered with Executive Services. This filename is used only when no filename is specified in the the command to query all system tasks.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 945 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_QueryAllTasksCmd().

13.2.1.132 CFE_PLATFORM_ES_ER_LOG_ENTRIES

```
#define CFE_PLATFORM_ES_ER_LOG_ENTRIES 20
```

Purpose Define Max Number of ER (Exception and Reset) log entries

Description:

Defines the maximum number of ER (Exception and Reset) log entries

Limits

There is a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of log entries is system dependent and should be verified.

Definition at line 554 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_ERLogDump(), and CFE_ES_WriteToERLog().

13.2.1.133 CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE

```
#define CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE 128
```

Purpose Maximum size of CPU Context in ES Error Log

Description:

This should be large enough to accommodate the CPU context information supplied by the PSP on the given platform.

Limits:

Must be greater than zero and a multiple of sizeof(uint32). Limited only by the available memory and the number of entries in the error log. Any context information beyond this size will be truncated.

Definition at line 568 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_WriteToERLog().

13.2.1.134 CFE_PLATFORM_ES_EXCEPTION_FUNCTION

```
#define CFE_PLATFORM_ES_EXCEPTION_FUNCTION CFE_ES_ProcessCoreException
```

Purpose Define cFE Core Exception Function

Description:

This parameter defines the function-to-call when a CPU or floating point exception occurs. The parameter is defaulted to call the ES API function CFE_ES_ProcessCoreException which handles the logging and reset from a system or cFE core exception.

Note: Exception interrupts are trapped at the Platform Support Package (PSP) layer. In order to initiate the cFE platform defined response to an exception, this platform defined callback function must be prototyped and called from the PSP exception hook API function CFE_PSP_ExceptionHook. For example:

– cfe_psp.h –

.... Prototype for exception ISR function implemented in CFE ....

typedef void (∗System_ExceptionFunc_t)(uint32 HostTaskId, const char ∗ReasonString, const uint32 ∗ContextPointer, uint32 ContextSize);

– cfe_pspexception.c –

.... Setup function pointer to CFE exception ISR callback ....

static const System_ExceptionFunc_t CFE_ExceptionCallback = CFE_PLATFORM_ES_EXCEPTION_FUNCTION;

void CFE_PSP_ExceptionHook (int task_id, int vector, uint8 ∗pEsf) { .... platform-specific logic ....

.... Use function pointer to call cFE routine to finish processing the exception ....

CFE_ExceptionCallback((uint32)task_id, CFE_PSP_ExceptionReasonString, (uint32 ∗)&CFE_PSP_ExceptionContext, sizeof(CFE_PSP_ExceptionContext_t));

}

Limits

Must be a valid function name.

Definition at line 1234 of file cpu1_platform_cfg.h.

13.2.1.135 CFE_PLATFORM_ES_MAX_APPLICATIONS

```
#define CFE_PLATFORM_ES_MAX_APPLICATIONS 32
```

Purpose Define Max Number of Applications

Description:

Defines the maximum number of applications that can be loaded into the system. This number does not include child tasks.

Limits

There is a lower limit of 6. The lower limit corresponds to the cFE internal applications. There are no restrictions on the upper limit however, the maximum number of applications is system dependent and should be verified. AppIDs that are checked against this configuration are defined by a 32 bit data word.

Definition at line 526 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CDS_ValidateAppID(), CFE_ES_CreateObjects(), CFE_ES_↩ DeleteChildTask(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfo(), CFE_ES_GetAppName(), CFE_ES_List↩ Applications(), CFE_ES_Main(), CFE_ES_MainTaskSyncDelay(), CFE_ES_QueryAllCmd(), CFE_ES_RestartApp(), CFE_ES_ScanAppTable(), CFE_EVS_CleanUpApp(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_EnableEvent↩ TypeCmd(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_SendEventWithAppID(), CFE_EVS_WriteAppData↩ FileCmd(), EVS_GetAppID(), EVS_GetApplicationInfo(), and EVS_SendEvent().

13.2.1.136 CFE_PLATFORM_ES_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_ES_MAX_BLOCK_SIZE 80000
```

Definition at line 1440 of file cpu1_platform_cfg.h.

13.2.1.137 CFE_PLATFORM_ES_MAX_GEN_COUNTERS

```
#define CFE_PLATFORM_ES_MAX_GEN_COUNTERS 8
```

Purpose Define Max Number of Generic Counters

Description:

Defines the maximum number of Generic Counters that can be registered.

Limits

This parameter has a lower limit of 1 and an upper limit of 65535.

Definition at line 611 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_DeleteGenCounter(), CFE_ES_GetGenCount(), CFE_ES_GetGenCounterIDByName(), CF↩ E_ES_IncrementGenCounter(), CFE_ES_Main(), CFE_ES_RegisterGenCounter(), and CFE_ES_SetGenCount().

13.2.1.138 CFE_PLATFORM_ES_MAX_LIBRARIES

```
#define CFE_PLATFORM_ES_MAX_LIBRARIES 10
```

Purpose Define Max Number of Shared libraries

Description:

Defines the maximum number of cFE Shared libraries that can be loaded into the system.

Limits

There is a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of libraries is system dependent and should be verified.

Definition at line 541 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_LoadLibrary().

13.2.1.139 CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS

```
#define CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS 2
```

Purpose Define Number of Processor Resets Before a Power On Reset

Description:

Number of Processor Resets before a Power On Reset is called. If set to 2, then 2 processor resets will occur, and the 3rd processor reset will be a power on reset instead.

Limits

There is a lower limit of 0. There are no restrictions on the upper limit however, the maximum number of processor resets may be system dependent and should be verified.

Definition at line 1403 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_SetupResetVariables().

13.2.1.140 CFE_PLATFORM_ES_MAX_SHELL_CMD

```
#define CFE_PLATFORM_ES_MAX_SHELL_CMD 64
```

Purpose Define Max Shell Command Size

Description:

Defines the maximum size in characters of the shell command.

Limits

There is a lower limit of 64 and an upper limit of OS_MAX_CMD_LEN. Units are characters.

Definition at line 881 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_ShellCmd().

13.2.1.141 CFE_PLATFORM_ES_MAX_SHELL_PKT

```
#define CFE_PLATFORM_ES_MAX_SHELL_PKT 64
```

Purpose Define Shell Command Telemetry Pkt Segment Size

Description:

Defines the size of the shell command tlm packet segments.The shell command output size is dependant on the shell command itself. If the shell output size is greater than the size of the packet defined here, the fsw will generate a series of tlm packets (of the size defined here) that can be reconstructed by the ground system.

Limits

There is a lower limit of 32 and an upper limit of CFE_SB_MAX_SB_MSG_SIZE.

Definition at line 897 of file cpu1_platform_cfg.h.

13.2.1.142 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 8
```

Purpose Define Default ES Memory Pool Block Sizes

Description:

Default Intermediate ES Memory Pool Block Sizes. If an application is using the CFE_ES Memory Pool APIs (CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_GetPoolBuf and CFE_ES_PutPoolBuf) but finds these sizes inappropriate for their use, they may wish to use the CFE_ES_PoolCreateEx API to specify their own intermediate block sizes

Limits

These sizes MUST be increasing and MUST be an integral multiple of 4. Also, CFE_PLATFORM_ES_MAX_BLOCK_SIZE must be larger than CFE_MISSION_SB_MAX_SB_MSG_SIZE and both CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE and CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE. Note that if Table Services have been removed from the CFE, the table size limits are still enforced although the table size definitions may be reduced. Refer to the CFS Deployment Guide for information about removing CFE Table Services from the CFE.

Definition at line 1424 of file cpu1_platform_cfg.h.

13.2.1.143 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 16
```

Definition at line 1425 of file cpu1_platform_cfg.h.

13.2.1.144 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 32
```

Definition at line 1426 of file cpu1_platform_cfg.h.

13.2.1.145 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 48
```

Definition at line 1427 of file cpu1_platform_cfg.h.

13.2.1.146 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 64
```

Definition at line 1428 of file cpu1_platform_cfg.h.

13.2.1.147 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 96
```

Definition at line 1429 of file cpu1_platform_cfg.h.

13.2.1.148 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 128
```

Definition at line 1430 of file cpu1_platform_cfg.h.

13.2.1.149 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08 160
```

Definition at line 1431 of file cpu1_platform_cfg.h.

13.2.1.150 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09 256
```

Definition at line 1432 of file cpu1_platform_cfg.h.

13.2.1.151 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10 512
```

Definition at line 1433 of file cpu1_platform_cfg.h.

13.2.1.152 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11 1024
```

Definition at line 1434 of file cpu1_platform_cfg.h.

13.2.1.153 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12 2048
```

Definition at line 1435 of file cpu1_platform_cfg.h.

13.2.1.154 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13 4096
```

Definition at line 1436 of file cpu1_platform_cfg.h.

13.2.1.155 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14 8192
```

Definition at line 1437 of file cpu1_platform_cfg.h.

13.2.1.156 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15 16384
```

Definition at line 1438 of file cpu1_platform_cfg.h.

13.2.1.157 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16

```
#define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16 32768
```

Definition at line 1439 of file cpu1_platform_cfg.h.

13.2.1.158 CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN

```
#define CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN 4
```

Purpose Define Memory Pool Alignment Size

Description:

Ensures that buffers obtained from a memory pool are aligned to a certain minimum block size. Note the allocator will always align to the minimum required by the CPU architecture. This may be set greater than the CPU requirement as desired for optimal performance.

For some architectures/applications it may be beneficial to set this to the cache line size of the target CPU, or to use special SIMD instructions that require a more stringent memory alignment.

Limits

This must always be a power of 2, as it is used as a binary address mask.

Definition at line 822 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_PoolCreateEx().

13.2.1.159 CFE_PLATFORM_ES_NONVOL_STARTUP_FILE

```
#define CFE_PLATFORM_ES_NONVOL_STARTUP_FILE "/cf/cfe_es_startup.scr"
```

Purpose ES Nonvolatile Startup Filename

Description:

The value of this constant defines the path and name of the file that contains a list of modules that will be loaded and started by the cFE after the cFE finishes its startup sequence.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 837 of file cpu1_platform_cfg.h.

13.2.1.160 CFE_PLATFORM_ES_OBJECT_TABLE_SIZE

```
#define CFE_PLATFORM_ES_OBJECT_TABLE_SIZE 30
```

Purpose Define Number of entries in the ES Object table

Description:

Defines the number of entries in the ES Object table. This table controls the core cFE startup.

Limits

There is a lower limit of 15. There are no restrictions on the upper limit however, the maximum object table size is system dependent and should be verified.

Definition at line 599 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_CreateObjects().

13.2.1.161 CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY

```
#define CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY 20
```

Purpose Define Performance Analyzer Child Task Delay

Description:

This parameter defines the delay time (in milliseconds) between performance data file writes performed by the Executive Services Performace Analyzer Child Task.

Limits

It is recommended this parameter be greater than or equal to 20ms. This parameter is limited by the maximum value allowed by the data type. In this case, the data type is an unsigned 32-bit integer, so the valid range is 0 to 0xFFFFFFFF.

Definition at line 1162 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_PerfLogDump(), and CFE_ES_StopPerfDataCmd().

13.2.1.162 CFE_PLATFORM_ES_PERF_CHILD_PRIORITY

```
#define CFE_PLATFORM_ES_PERF_CHILD_PRIORITY 200
```

Purpose Define Performance Analyzer Child Task Priority

Description:

This parameter defines the priority of the child task spawed by the Executive Services to write performance data to a file. Lower numbers are higher priority, with 1 being the highest priority in the case of a child task.

Limits

Valid range for a child task is 1 to 255 however, the priority cannot be higher (lower number) than the ES parent application priority.

Definition at line 1133 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.2.1.163 CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE

```
#define CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE 4096
```

Purpose Define Performance Analyzer Child Task Stack Size

Description:

This parameter defines the stack size of the child task spawed by the Executive Services to write performance data to a file.

Limits

It is recommended this parameter be greater than or equal to 4KB. This parameter is limited by the maximum value allowed by the data type. In this case, the data type is an unsigned 32-bit integer, so the valid range is 0 to 0xFFFFFFFF.

Definition at line 1147 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.2.1.164 CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE

```
#define CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE 10000
```

Purpose Define Max Size of Performance Data Buffer

Description:

Defines the maximum size of the performance data buffer. Units are number of performance data entries. An entry is defined by a 32 bit data word followed by a 64 bit time stamp.

Limits

There is a lower limit of 1025. There are no restrictions on the upper limit however, the maximum buffer size size is system dependent and should be verified. The units are number of entries. An entry is defined by a 32 bit data word followed by a 64 bit time stamp.

Definition at line 1052 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_PerfLogAdd().

13.2.1.165 CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS

```
#define CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS 50
```

Purpose Define Performance Analyzer Child Task Number of Entries Between Delay

Description:

This parameter defines the number of performace analyzer entries the Performace Analyzer Child Task will write to the file between delays.

Definition at line 1172 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_PerfLogDump(), and CFE_ES_StopPerfDataCmd().

13.2.1.166 CFE_PLATFORM_ES_PERF_FILTMASK_ALL

```
#define CFE_PLATFORM_ES_PERF_FILTMASK_ALL ~CFE_PLATFORM_ES_PERF_FILTMASK_NONE
```

Purpose Define Filter Mask Setting for Enabling All Performance Entries

Description:

Defines the filter mask for enabling all performance entries. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 1073 of file cpu1_platform_cfg.h.

13.2.1.167 CFE_PLATFORM_ES_PERF_FILTMASK_INIT

```
#define CFE_PLATFORM_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_PERF_FILTMASK_ALL
```

Purpose Define Default Filter Mask Setting for Performance Data Buffer

Description:

Defines the default filter mask for the performance data buffer. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 1084 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_SetupPerfVariables().

13.2.1.168 CFE_PLATFORM_ES_PERF_FILTMASK_NONE

```
#define CFE_PLATFORM_ES_PERF_FILTMASK_NONE 0
```

Purpose Define Filter Mask Setting for Disabling All Performance Entries

Description:

Defines the filter mask for disabling all performance entries. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 1063 of file cpu1_platform_cfg.h.

13.2.1.169 CFE_PLATFORM_ES_PERF_MAX_IDS

```
#define CFE_PLATFORM_ES_PERF_MAX_IDS 128
```

Purpose Define Max Number of Performance IDs

Description:

Defines the maximum number of perf ids allowed.

Limits

This number must always be divisible by 32. There is a lower limit of 32 and an upper limit of 512 on this configuration paramater.

Definition at line 1036 of file cpu1_platform_cfg.h.

13.2.1.170 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL

```
#define CFE_PLATFORM_ES_PERF_TRIGMASK_ALL ~CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

Purpose Define Filter Trigger Setting for Enabling All Performance Entries

Description:

Defines the trigger mask for enabling all performance data entries. The value is a bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 1107 of file cpu1_platform_cfg.h.

13.2.1.171 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT

```
#define CFE_PLATFORM_ES_PERF_TRIGMASK_INIT CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

Purpose Define Default Filter Trigger Setting for Performance Data Buffer

Description:

Defines the default trigger mask for the performance data buffer. The value is a 32-bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 1118 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_SetupPerfVariables().

13.2.1.172 CFE_PLATFORM_ES_PERF_TRIGMASK_NONE

```
#define CFE_PLATFORM_ES_PERF_TRIGMASK_NONE 0
```

Purpose Define Default Filter Trigger Setting for Disabling All Performance Entries

Description:

Defines the default trigger mask for disabling all performance data entries. The value is a bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 1096 of file cpu1_platform_cfg.h.

13.2.1.173 CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING

```
#define CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING "/ram"
```

Purpose RAM Disk Mount string

Description:

The CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING parameter is used to set the cFE mount path for the CFE RAM disk. This is a parameter for missions that do not want to use the default value of "/ram", or for missions that need to have a different value for different CPUs or Spacecraft. Note that the vxWorks OSAL cannot currently handle names that have more than one path separator in it. The names "/ram", "/ramdisk", "/disk123" will all work, but "/disks/ram" will not. Multiple separators can be used with the posix or RTEMS ports.

Definition at line 740 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_InitializeFileSystems(), and CFE_ES_LoadLibrary().

13.2.1.174 CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS

```
#define CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS 4096
```

Purpose ES Ram Disk Number of Sectors

Description:

Defines the ram disk number of sectors. The ram disk is one of four memory areas that are preserved on a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum number of RAM sectors is system dependent and should be verified.

Definition at line 699 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_InitializeFileSystems().

13.2.1.175 CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED

```
#define CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED 30
```

Purpose Percentage of Ram Disk Reserved for Decompressing Apps

Description:

The CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED parameter is used to make sure that the Volatile (RAM) Disk has a defined amount of free space during a processor reset. The cFE uses the Volatile disk to decompress cFE applications during system startup. If this Volatile disk happens to get filled with logs and misc files, then a processor reset may not work, because there will be no room to decompress cFE apps. To solve that problem, this parameter sets the "Low Water Mark" for disk space on a Processor reset. It should be set to allow the largest cFE Application to be decompressed. During a Processor reset, if there is not sufficient space left on the disk, it will be re-formatted in order to clear up some space.

This feature can be turned OFF by setting the parameter to 0.

Limits

There is a lower limit of 0 and an upper limit of 75 on this configuration paramater.Units are percentage. A setting of zero will turn this feature off.

Definition at line 723 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_InitializeFileSystems().

13.2.1.176 CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE

```
#define CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE 512
```

Purpose ES Ram Disk Sector Size

Description:

Defines the ram disk sector size. The ram disk is 1 of 4 memory areas that are preserved on a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in C↩ FE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum RAM disk sector size is system dependent and should be verified.

Definition at line 680 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_InitializeFileSystems().

13.2.1.177 CFE_PLATFORM_ES_RESET_AREA_SIZE

```
#define CFE_PLATFORM_ES_RESET_AREA_SIZE ( 170 * 1024 )
```

Purpose Define ES Reset Area Size

Description:

The ES Reset Area Size. This is the size in bytes of the cFE Reset variable and log area. This is a block of memory used by the cFE to store the system log ER Log and critical reset variables. This is 4 of 4 of the memory areas that are preserved during a processor reset. Note: This area must be sized large enough to hold all of the data structures. It should be automatically sized based on the CFE_ES_ResetData_t type, but circular dependancies in the headers prevent it from being defined this way. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 153600 (150KBytes) and an upper limit of UINT_MAX (4 Gigabytes) on this configuration paramater.

Definition at line 803 of file cpu1_platform_cfg.h.

13.2.1.178 CFE_PLATFORM_ES_SHELL_OS_DELAY_MILLISEC

```
#define CFE_PLATFORM_ES_SHELL_OS_DELAY_MILLISEC 200
```

Purpose Define OS Task Delay Value for ES Shell Command

Description:

This parameter defines the length of time (in milliseconds) ES will delay when sending shell command packets over the software bus to not flood the pipe on large messages.

Note: The milliseconds passed into OS_TaskDelay are converted into the units the underlying OS uses to measure time passing. Many platforms limit the precision of this value however, a delay may not be needed at all in which the value may be set to zero.

Limits

Not Applicable

Definition at line 915 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_ShellOutputCommand().

13.2.1.179 CFE_PLATFORM_ES_START_TASK_PRIORITY

```
#define CFE_PLATFORM_ES_START_TASK_PRIORITY 68
```

Purpose Define ES Task Priority

Description:

Defines the cFE_ES Task priority.

Limits

Not Applicable

Definition at line 1297 of file cpu1_platform_cfg.h.

13.2.1.180 CFE_PLATFORM_ES_START_TASK_STACK_SIZE

```
#define CFE_PLATFORM_ES_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Purpose Define ES Task Stack Size

Description:

Defines the cFE_ES Task Stack Size

Limits

There is a lower limit of 2048 on this configuration paramater. There are no restrictions on the upper limit however, the maximum stack size size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1312 of file cpu1_platform_cfg.h.

13.2.1.181 CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC

```
#define CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC 1000
```

Purpose Startup script timeout

Description:

The upper limit for the total amount of time that all apps listed in the CFE ES startup script may take to all become ready.

Unlike the "core" app timeout, this is a soft limit; if the alloted time is exceeded, it probably indicates an issue with one of the apps, but does not cause CFE ES to take any additional action other than logging the event to the syslog.

Units are in milliseconds

Limits:

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 1894 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_Main().

13.2.1.182 CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC

```
#define CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC 50
```

Purpose Poll timer for startup sync delay

Description:

During startup, some tasks may need to synchronize their own initialization with the initialization of other applications in the system.

CFE ES implements an API to accomplish this, that performs a task delay (sleep) while polling the overall system state until other tasks are ready.

This value controls the amount of time that the CFE_ES_ApplicationSyncDelay will sleep between each check of the system state. This should be large enough to allow other tasks to run, but not so large as to noticeably delay the startup completion.

Units are in milliseconds

Limits:

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 1852 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_MainTaskSyncDelay(), and CFE_ES_WaitForSystemState().

13.2.1.183 CFE_PLATFORM_ES_SYSTEM_LOG_SIZE

```
#define CFE_PLATFORM_ES_SYSTEM_LOG_SIZE 3072
```

Purpose Define Size of the cFE System Log.

Description:

Defines the size in bytes of the cFE system log. The system log holds variable length strings that are terminated by a linefeed and null character.

Limits

There is a lower limit of 512. There are no restrictions on the upper limit however, the maximum system log size is system dependent and should be verified.

Definition at line 584 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_HousekeepingCmd(), and CFE_ES_SysLogAppend_Unsync().

13.2.1.184 CFE_PLATFORM_ES_USER_RESERVED_SIZE

```
#define CFE_PLATFORM_ES_USER_RESERVED_SIZE ( 1024 * 1024 )
```

Purpose Define User Reserved Memory Size

Description:

User Reserved Memory Size. This is the size in bytes of the cFE User reserved Memory area. This is a block of memory that is available for cFE application use. The address is obtained by calling CFE_PSP_GetUserReservedArea. The User Reserved Memory is one of four memory areas that are preserved during a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE_PSP) such as USER_RESERVED_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 1024 and an upper limit of UINT_MAX (4 Gigabytes) on this configuration paramater.

Definition at line 779 of file cpu1_platform_cfg.h.

13.2.1.185 CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE

```
#define CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE "/ram/cfe_es_startup.scr"
```

Purpose ES Volatile Startup Filename

Description:

The value of this constant defines the path and name of the file that contains a list of modules that will be loaded and started by the cFE after the cFE finishes its startup sequence.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 852 of file cpu1_platform_cfg.h.

Referenced by CFE_ES_StartApplications().

13.2.1.186 CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE

```
#define CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE "/ram/cfe_evs_app.dat"
```

Purpose Default EVS Application Data Filename

Description:

The value of this constant defines the filename used to store the EVS Application Data(event counts/filtering information). This filename is used only when no filename is specified in the command to dump the event log.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 1541 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

13.2.1.187 CFE_PLATFORM_EVS_DEFAULT_LOG_FILE

```
#define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE "/ram/cfe_evs.log"
```

Purpose Default Event Log Filename

Description:

The value of this constant defines the filename used to store the Event Services local event log. This filename is used only when no filename is specified in the command to dump the event log.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 1512 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

13.2.1.188 CFE_PLATFORM_EVS_DEFAULT_LOG_MODE

```
#define CFE_PLATFORM_EVS_DEFAULT_LOG_MODE 1
```

Purpose Default EVS Local Event Log Mode

Description:

Defines a state of overwrite(0) or discard(1) for the operation of the EVS local event log. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest event in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. Overwrite Mode = 0, Discard Mode = 1.

Limits

The valid settings are 0 or 1

Definition at line 1592 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_EarlyInit().

13.2.1.189 CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE

```
#define CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE CFE_EVS_MsgFormat_LONG
```

Purpose Default EVS Message Format Mode

Description:

Defines the default message format (long or short) for event messages being sent to the ground. Choose between CFE_EVS_MsgFormat_LONG or CFE_EVS_MsgFormat_SHORT.

Limits

The valid settings are CFE_EVS_MsgFormat_LONG or CFE_EVS_MsgFormat_SHORT

Definition at line 1606 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_EarlyInit().

13.2.1.190 CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG

```
#define CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG 0xE
```

Purpose Default EVS Event Type Filter Mask

Description:

Defines a state of on or off for all four event types. The term event 'type' refers to the criticality level and may be Debug, Informational, Error or Critical. Each event type has a bit position. (bit 0 = Debug, bit 1 = Info, bit 2 = Error, bit 3 = Critical). This is a global setting, meaning it applies to all applications. To filter an event type, set its bit to zero. For example, 0xE means Debug = OFF, Info = ON, Error = ON, Critical = ON

Limits

The valid settings are 0x0 to 0xF.

Definition at line 1574 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_Register().

13.2.1.191 CFE_PLATFORM_EVS_LOG_MAX

```
#define CFE_PLATFORM_EVS_LOG_MAX 20
```

Purpose Maximum Number of Events in EVS Local Event Log

Description:

Dictates the EVS local event log capacity. Units are the number of events.

Limits

There are no restrictions on the lower and upper limits however, the maximum log size is system dependent and should be verified.

Definition at line 1525 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_EarlyInit(), CFE_EVS_WriteLogDataFileCmd(), and EVS_AddLog().

13.2.1.192 CFE_PLATFORM_EVS_LOG_ON

```
#define CFE_PLATFORM_EVS_LOG_ON
```

Purpose Enable or Disable EVS Local Event Log

Description:

The CFE_PLATFORM_EVS_LOG_ON configuration parameter must be defined to enable EVS event logging. In order to disable the local event log this definition needs to be commented out.

Limits

Not Applicable

Definition at line 1497 of file cpu1_platform_cfg.h.

13.2.1.193 CFE_PLATFORM_EVS_MAX_EVENT_FILTERS

```
#define CFE_PLATFORM_EVS_MAX_EVENT_FILTERS 8
```

Purpose Define Maximum Number of Event Filters per Application

Description:

Maximum number of events that may be filtered per application.

Limits

There are no restrictions on the lower and upper limits however, the maximum number of event filters is system dependent and should be verified.

Definition at line 1483 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_↩ ResetAllFiltersCmd(), CFE_EVS_WriteAppDataFileCmd(), CFE_SB_AppInit(), and EVS_FindEventID().

13.2.1.194 CFE_PLATFORM_EVS_PORT_DEFAULT

```
#define CFE_PLATFORM_EVS_PORT_DEFAULT 0x0001
```

Purpose Default EVS Output Port State

Description:

Defines the default port state (enabled or deisabled) for the four output ports defined within the Event Service. Port 1 is usually the uart output terminal. To enable a port, set the proper bit to a 1. Bit 0 is port 1, bit 1 is port2 etc.

Limits

The valid settings are 0x0 to 0xF.

Definition at line 1556 of file cpu1_platform_cfg.h.

Referenced by CFE_EVS_EarlyInit().

13.2.1.195 CFE_PLATFORM_EVS_START_TASK_PRIORITY

```
#define CFE_PLATFORM_EVS_START_TASK_PRIORITY 61
```

Purpose Define EVS Task Priority

Description:

Defines the cFE_EVS Task priority.

Limits

Not Applicable

Definition at line 1245 of file cpu1_platform_cfg.h.

13.2.1.196 CFE_PLATFORM_EVS_START_TASK_STACK_SIZE

```
#define CFE_PLATFORM_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Purpose Define EVS Task Stack Size

Description:

Defines the cFE_EVS Task Stack Size

Limits

There is a lower limit of 2048 on this configuration paramater. There are no restrictions on the upper limit however, the maximum stack size size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1260 of file cpu1_platform_cfg.h.

13.2.1.197 CFE_PLATFORM_SB_BUF_MEMORY_BYTES

```
#define CFE_PLATFORM_SB_BUF_MEMORY_BYTES 524288
```

Purpose Size of the SB buffer memory pool

Description:

Dictates the size of the SB memory pool. For each message the SB sends, the SB dynamically allocates from this memory pool, the memory needed to process the message. The memory needed to process each message is msg size + msg descriptor(CFE_SB_BufferD_t). This memory pool is also used to allocate destination descriptors (CFE_SB_DestinationD_t) during the subscription process. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'. Some memory statistics have been added to the SB housekeeping packet. NOTE: It is important to monitor these statistics to ensure the desired memory margin is met.

Limits

This parameter has a lower limit of 512 and an upper limit of UINT_MAX (4 Gigabytes).

Definition at line 142 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit(), CFE_SB_InitBuffers(), and CFE_SB_SendHKTlmCmd().

13.2.1.198 CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME

```
#define CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME "/ram/cfe_sb_msgmap.dat"
```

Purpose Default Message Map Filename

Description:

The value of this constant defines the filename used to store the software bus message map information. This filename is used only when no filename is specified in the command. The message map is a lookup table (array of 16bit words) that has an element for each possible MsgId value and holds the routing table index for that MsgId. The Msg Map provides fast access to the destinations of a message.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 241 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_SendMapInfoCmd().

13.2.1.199 CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT

```
#define CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT 4
```

Purpose Default Subscription Message Limit

Description:

Dictates the default Message Limit when using the CFE_SB_Subscribe API. This will limit the number of messages with a specific message ID that can be received through a subscription. This only changes the default; other message limits can be set on a per subscription basis using CFE_SB_SubscribeEx .

Limits

This parameter has a lower limit of 4 and an upper limit of 65535.

Definition at line 119 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_Subscribe().

13.2.1.200 CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME

```
#define CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME "/ram/cfe_sb_pipe.dat"
```

Purpose Default Pipe Information Filename

Description:

The value of this constant defines the filename used to store the software bus pipe information. This filename is used only when no filename is specified in the command.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 223 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_SendPipeInfoCmd().

13.2.1.201 CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER

```
#define CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER 1
```

Purpose Define Default Sender Information Storage Mode

Description:

Defines the default mode for the storing of sender information when sending a software bus message. If set to 1, the sender information will be stored. If set to 0, the sender information will not be stored.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration paramater.

Definition at line 325 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_EarlyInit().

13.2.1.202 CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME

```
#define CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME "/ram/cfe_sb_route.dat"
```

Purpose Default Routing Information Filename

Description:

The value of this constant defines the filename used to store the software bus routing information. This filename is used only when no filename is specified in the command.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 208 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_SendRoutingInfoCmd().

13.2.1.203 CFE_PLATFORM_SB_FILTER_MASK1

```
#define CFE_PLATFORM_SB_FILTER_MASK1 CFE_EVS_FIRST_4_STOP
```

Definition at line 260 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.204 CFE_PLATFORM_SB_FILTER_MASK2

```
#define CFE_PLATFORM_SB_FILTER_MASK2 CFE_EVS_FIRST_4_STOP
```

Definition at line 263 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.205 CFE_PLATFORM_SB_FILTER_MASK3

```
#define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
```

Definition at line 266 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.206 CFE_PLATFORM_SB_FILTER_MASK4

```
#define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
```

Definition at line 269 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.207 CFE_PLATFORM_SB_FILTER_MASK5

```
#define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
```

Definition at line 272 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.208 CFE_PLATFORM_SB_FILTER_MASK6

```
#define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
```

Definition at line 275 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.209 CFE_PLATFORM_SB_FILTER_MASK7

```
#define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
```

Definition at line 278 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.210 CFE_PLATFORM_SB_FILTER_MASK8

```
#define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
```

Definition at line 281 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.211 CFE_PLATFORM_SB_FILTERED_EVENT1

```
#define CFE_PLATFORM_SB_FILTERED_EVENT1 CFE_SB_SEND_NO_SUBS_EID
```

Purpose SB Event Filtering

Description:

This group of configuration paramters dictates what SB events will be filtered through EVS. The filtering will begin after the SB task initializes and stay in effect until a cmd to EVS changes it. This allows the operator to set limits on the number of event messages that are sent during system initialization. NOTE: Set all unused event values and mask values to zero

Limits

This filtering applies only to SB events. These parameters have a lower limit of 0 and an upper limit of 65535.

Definition at line 259 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.212 CFE_PLATFORM_SB_FILTERED_EVENT2

```
#define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
```

Definition at line 262 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.213 CFE_PLATFORM_SB_FILTERED_EVENT3

```
#define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
```

Definition at line 265 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.214 CFE_PLATFORM_SB_FILTERED_EVENT4

```
#define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
```

Definition at line 268 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.215 CFE_PLATFORM_SB_FILTERED_EVENT5

```
#define CFE_PLATFORM_SB_FILTERED_EVENT5 0
```

Definition at line 271 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.216 CFE_PLATFORM_SB_FILTERED_EVENT6

```
#define CFE_PLATFORM_SB_FILTERED_EVENT6 0
```

Definition at line 274 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.217 CFE_PLATFORM_SB_FILTERED_EVENT7

```
#define CFE_PLATFORM_SB_FILTERED_EVENT7 0
```

Definition at line 277 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.218 CFE_PLATFORM_SB_FILTERED_EVENT8

```
#define CFE_PLATFORM_SB_FILTERED_EVENT8 0
```

Definition at line 280 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit().

13.2.1.219 CFE_PLATFORM_SB_HIGHEST_VALID_MSGID

```
#define CFE_PLATFORM_SB_HIGHEST_VALID_MSGID 0x1FFF
```

Purpose Highest Valid Message Id

Description:

The value of this constant dictates the size of the SB message map. The SB message map is a lookup table that provides the routing table index for fast access into the routing table. The default setting of 0x1FFF was chosen to save memory. This reduces the message map from 128Kbytes to 16Kbytes. See CFE_FSW_DCR 504 for more details.

If this value is different in a distributed architecture some platforms may not be able to subscribe to messages generated on other platforms since the message id would exceed the mapping table's highest index. Care would have to be taken to ensure the constrained platform did not subscribe to message Ids that exceed CFE_PLATFORM_SB_HIGHEST_↩ VALID_MSGID

The recommended case to to have this value the same across all mission platforms

Limits

This parameter has a lower limit of 1 and an upper limit of 0xFFFF.

Definition at line 183 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_IsValidMsgId().

13.2.1.220 CFE_PLATFORM_SB_MAX_BLOCK_SIZE

```
#define CFE_PLATFORM_SB_MAX_BLOCK_SIZE (CFE_MISSION_SB_MAX_SB_MSG_SIZE + 40)
```

Definition at line 311 of file cpu1_platform_cfg.h.

13.2.1.221 CFE_PLATFORM_SB_MAX_DEST_PER_PKT

```
#define CFE_PLATFORM_SB_MAX_DEST_PER_PKT 16
```

Purpose Maximum Number of unique local destinations a single MsgId can have

Description:

Dictates the maximum number of unique local destinations a single MsgId can have.

Limits

This parameter has a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of destinations per packet is system dependent and should be verified. Destination number values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 103 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_SubscribeFull().

13.2.1.222 CFE_PLATFORM_SB_MAX_MSG_IDS

```
#define CFE_PLATFORM_SB_MAX_MSG_IDS 256
```

Purpose Maximum Number of Unique Message IDs SB Routing Table can hold

Description:

Dictates the maximum number of unique MsgIds the SB routing table will hold. This constant has a direct affect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

Limits

This parameter has a lower limit of 1 and an upper limit of 1024.

Definition at line 68 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit(), CFE_SB_DeletePipeFull(), CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_InitIdx↩Stack(), CFE_SB_InitRoutingTbl(), CFE_SB_IsValidRouteIdx(), CFE_SB_RouteIdxPop_Unsync(), CFE_SB_Send↩PrevSubsCmd(), and CFE_SB_SubscribeFull().

13.2.1.223 CFE_PLATFORM_SB_MAX_PIPE_DEPTH

```
#define CFE_PLATFORM_SB_MAX_PIPE_DEPTH 256
```

Purpose Maximum depth allowed when creating an SB pipe

Description:

The value of this constant dictates the maximum pipe depth that an application may request. The pipe depth is given as a paramter in the CFE_SB_CreatePipe API.

Limits

This parameter has a lower limit of 1. There are no restrictions on the upper limit however, the maximum pipe depth is system dependent and should be verified. Pipe Depth values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 159 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit(), and CFE_SB_CreatePipe().

13.2.1.224 CFE_PLATFORM_SB_MAX_PIPES

```
#define CFE_PLATFORM_SB_MAX_PIPES 64
```

Purpose Maximum Number of Unique Pipes SB Routing Table can hold

Description:

Dictates the maximum number of unique Pipes the SB routing table will hold. This constant has a direct affect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

Limits

This parameter has a lower limit of 1. This parameter must also be less than or equal to OS_MAX_QUEUES.

Definition at line 86 of file cpu1_platform_cfg.h.

Referenced by CFE_SB_AppInit(), CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_GetAvailPipeIdx(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeName(), CFE_SB_InitPipeTbl(), CFE_SB_SendPipeInfo(), and CFE_SB_ValidatePipeId().

13.2.1.225 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01 8
```

Purpose Define SB Memory Pool Block Sizes

Description:

Software Bus Memory Pool Block Sizes

Limits

These sizes MUST be increasing and MUST be an integral multiple of 4. The number of block sizes defined cannot exceed CFE_ES_MAX_MEMPOOL_BLOCK_SIZES

Definition at line 295 of file cpu1_platform_cfg.h.

13.2.1.226 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16
```

Definition at line 296 of file cpu1_platform_cfg.h.

13.2.1.227 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20
```

Definition at line 297 of file cpu1_platform_cfg.h.

13.2.1.228 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36
```

Definition at line 298 of file cpu1_platform_cfg.h.

13.2.1.229 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64
```

Definition at line 299 of file cpu1_platform_cfg.h.

13.2.1.230 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96
```

Definition at line 300 of file cpu1_platform_cfg.h.

13.2.1.231 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128
```

Definition at line 301 of file cpu1_platform_cfg.h.

13.2.1.232 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160
```

Definition at line 302 of file cpu1_platform_cfg.h.

13.2.1.233 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256
```

Definition at line 303 of file cpu1_platform_cfg.h.

13.2.1.234 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
```

Definition at line 304 of file cpu1_platform_cfg.h.

13.2.1.235 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
```

Definition at line 305 of file cpu1_platform_cfg.h.

13.2.1.236 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
```

Definition at line 306 of file cpu1_platform_cfg.h.

13.2.1.237 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
```

Definition at line 307 of file cpu1_platform_cfg.h.

13.2.1.238 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
```

Definition at line 308 of file cpu1_platform_cfg.h.

13.2.1.239 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
```

Definition at line 309 of file cpu1_platform_cfg.h.

13.2.1.240 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16

```
#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
```

Definition at line 310 of file cpu1_platform_cfg.h.

13.2.1.241 CFE_PLATFORM_SB_START_TASK_PRIORITY

```
#define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
```

Purpose Define SB Task Priority

Description:

Defines the cFE_SB Task priority.

Limits

Not Applicable

Definition at line 1271 of file cpu1_platform_cfg.h.

13.2.1.242 CFE_PLATFORM_SB_START_TASK_STACK_SIZE

```
#define CFE_PLATFORM_SB_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Purpose Define SB Task Stack Size

Description:

Defines the cFE_SB Task Stack Size

Limits

There is a lower limit of 2048 on this configuration paramater. There are no restrictions on the upper limit however, the maximum stack size size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1286 of file cpu1_platform_cfg.h.

13.2.1.243 CFE_PLATFORM_TBL_BUF_MEMORY_BYTES

```
#define CFE_PLATFORM_TBL_BUF_MEMORY_BYTES 524288
```

Purpose Size of Table Services Table Memory Pool

Description:

Defines the TOTAL size of the memory pool that cFE Table Services allocates from the system. The size must be large enough to provide memory for each registered table, the inactive buffers for double buffered tables and for the shared inactive buffers for single buffered tables.

Limits

The cFE does not place a limit on the size of this parameter.

Definition at line 1624 of file cpu1_platform_cfg.h.

13.2.1.244 CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE

```
#define CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE "/ram/cfe_tbl_reg.log"
```

Purpose Default Filename for a Table Registry Dump

Description:

Defines the file name used to store the table registry when no filename is specified in the dump registry command.

Limits

The length of each string, including the NULL terminator cannot exceed the OS_MAX_PATH_LEN value.

Definition at line 1738 of file cpu1_platform_cfg.h.

13.2.1.245 CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES

```
#define CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES 32
```

Purpose Maximum Number of Critical Tables that can be Registered

Description:

Defines the maximum number of critical tables supported by this processor's Table Services.

Limits

This number must be less than 32767. It should be recognized that this parameter determines the size of the Critical Table Registry which is maintained in the Critical Data Store. An excessively high number will waste Critical Data Store memory. Therefore, this number must not exceed the value defined in CFE_ES_CDS_MAX_CRITICAL_T↩ ABLES.

Definition at line 1679 of file cpu1_platform_cfg.h.

13.2.1.246 CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE

```
#define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE 16384
```

Purpose Maximum Size Allowed for a Double Buffered Table

Description:

Defines the maximum allowed size (in bytes) of a double buffered table.

Limits

The cFE does not place a limit on the size of this parameter but it must be less than half of CFE_PLATFORM_T↩ BL_BUF_MEMORY_BYTES.

Definition at line 1636 of file cpu1_platform_cfg.h.

13.2.1.247 CFE_PLATFORM_TBL_MAX_NUM_HANDLES

```
#define CFE_PLATFORM_TBL_MAX_NUM_HANDLES 256
```

Purpose Maximum Number of Table Handles

Description:

Defines the maximum number of Table Handles.

Limits

This number must be less than 32767. This number must be at least as big as the number of tables (CFE_PLAT↩ FORM_TBL_MAX_NUM_TABLES) and should be set higher if tables are shared between applications.

Definition at line 1692 of file cpu1_platform_cfg.h.

13.2.1.248 CFE_PLATFORM_TBL_MAX_NUM_TABLES

```
#define CFE_PLATFORM_TBL_MAX_NUM_TABLES 128
```

Purpose Maximum Number of Tables Allowed to be Registered

Description:

Defines the maximum number of tables supported by this processor's Table Services.

Limits

This number must be less than 32767. It should be recognized that this parameter determines the size of the Table Registry. An excessively high number will waste memory.

Definition at line 1665 of file cpu1_platform_cfg.h.

13.2.1.249 CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS

```
#define CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS 10
```

Purpose Maximum Number of Simultaneous Table Validations

Description:

Defines the maximum number of pending validations that the Table Services can handle at any one time. When a table has a validation function, a validation request is made of the application to perform that validation. This number determines how many of those requests can be outstanding at any one time.

Limits

This number must be less than 32767. An excessively high number will degrade system performance and waste memory. A number less than 20 is suggested but not required.

Definition at line 1725 of file cpu1_platform_cfg.h.

13.2.1.250 CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS

```
#define CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS 4
```

Purpose Maximum Number of Simultaneous Loads to Support

Description:

Defines the maximum number of single buffered tables that can be loaded simultaneously. This number is used to determine the number of shared buffers to allocate.

Limits

This number must be less than 32767. An excessively high number will degrade system performance and waste memory. A number less than 5 is suggested but not required.

Definition at line 1707 of file cpu1_platform_cfg.h.

13.2.1.251 CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE

```
#define CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE 16384
```

Purpose Maximum Size Allowed for a Single Buffered Table

Description:

Defines the maximum allowed size (in bytes) of a single buffered table. **NOTE:** This size determines the size of all shared table buffers. Therefore, this size will be multiplied by CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_L↩OADS below when allocating memory for shared tables.

Limits

The cFE does not place a limit on the size of this parameter but it must be small enough to allow for CFE_PLATF↩ORM_TBL_MAX_SIMULTANEOUS_LOADS number of tables to fit into CFE_PLATFORM_TBL_BUF_MEMOR↩Y_BYTES.

Definition at line 1652 of file cpu1_platform_cfg.h.

13.2.1.252 CFE_PLATFORM_TBL_START_TASK_PRIORITY

```
#define CFE_PLATFORM_TBL_START_TASK_PRIORITY 70
```

Purpose Define TBL Task Priority

Description:

Defines the cFE_TBL Task priority.

Limits

Not Applicable

Definition at line 1359 of file cpu1_platform_cfg.h.

13.2.1.253 CFE_PLATFORM_TBL_START_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Purpose Define TBL Task Stack Size

Description:

Defines the cFE_TBL Task Stack Size

Limits

There is a lower limit of 2048 on this configuration paramater. There are no restrictions on the upper limit however, the maximum stack size size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1374 of file cpu1_platform_cfg.h.

13.2.1.254 CFE_PLATFORM_TBL_U32FROM4CHARS

```
#define CFE_PLATFORM_TBL_U32FROM4CHARS(
              _C1,
              _C2,
              _C3,
              _C4 )
```

Value:

```
( (uint32)(_C1) << 24 | \
   (uint32)(_C2) << 16 | \
   (uint32)(_C3) << 8 | \
   (uint32)(_C4) )
```

Definition at line 1760 of file cpu1_platform_cfg.h.

13.2.1.255 CFE_PLATFORM_TBL_VALID_PRID_1

```
#define CFE_PLATFORM_TBL_VALID_PRID_1 (CFE_PLATFORM_CPU_ID)
```

Purpose Processor ID values used for table load validation

Description:

Defines the processor ID values used for validating the processor ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

Limits

This value can be any 32 bit unsigned integer.

Definition at line 1812 of file cpu1_platform_cfg.h.

13.2.1.256 CFE_PLATFORM_TBL_VALID_PRID_2

```
#define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
```

Definition at line 1813 of file cpu1_platform_cfg.h.

13.2.1.257 CFE_PLATFORM_TBL_VALID_PRID_3

```
#define CFE_PLATFORM_TBL_VALID_PRID_3 0
```

Definition at line 1814 of file cpu1_platform_cfg.h.

13.2.1.258 CFE_PLATFORM_TBL_VALID_PRID_4

```
#define CFE_PLATFORM_TBL_VALID_PRID_4 0
```

Definition at line 1815 of file cpu1_platform_cfg.h.

13.2.1.259 CFE_PLATFORM_TBL_VALID_PRID_COUNT

```
#define CFE_PLATFORM_TBL_VALID_PRID_COUNT 0
```

Purpose Number of Processor ID's specified for validation

Description:

Defines the number of specified processor ID values that are verified during table loads. If the number is zero then no validation of the processor ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of processor ID's defined below are compared to the processor ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified processor ID values.

Limits

This number must be greater than or equal to zero and less than or equal to 4.

Definition at line 1798 of file cpu1_platform_cfg.h.

13.2.1.260 CFE_PLATFORM_TBL_VALID_SCID_1

```
#define CFE_PLATFORM_TBL_VALID_SCID_1 (CFE_MISSION_SPACECRAFT_ID)
```

Purpose Spacecraft ID values used for table load validation

Description:

Defines the spacecraft ID values used for validating the spacecraft ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

Limits

This value can be any 32 bit unsigned integer.

Definition at line 1778 of file cpu1_platform_cfg.h.

13.2.1.261 CFE_PLATFORM_TBL_VALID_SCID_2

```
#define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
```

Definition at line 1779 of file cpu1_platform_cfg.h.

13.2.1.262 CFE_PLATFORM_TBL_VALID_SCID_COUNT

```
#define CFE_PLATFORM_TBL_VALID_SCID_COUNT 0
```

Purpose Number of Spacecraft ID's specified for validation

Description:

Defines the number of specified spacecraft ID values that are verified during table loads. If the number is zero then no validation of the spacecraft ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of spacecraft ID's defined below are compared to the spacecraft ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified spacecraft ID values.

Limits

This number must be greater than or equal to zero and less than or equal to 2.

Definition at line 1757 of file cpu1_platform_cfg.h.

13.2.1.263 CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY 25
```

Definition at line 1329 of file cpu1_platform_cfg.h.

13.2.1.264 CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE 8192
```

Definition at line 1348 of file cpu1_platform_cfg.h.

13.2.1.265 CFE_PLATFORM_TIME_CFG_CLIENT

```
#define CFE_PLATFORM_TIME_CFG_CLIENT false
```

Definition at line 341 of file cpu1_platform_cfg.h.

13.2.1.266 CFE_PLATFORM_TIME_CFG_LATCH_FLY

```
#define CFE_PLATFORM_TIME_CFG_LATCH_FLY 8
```

Purpose Define Periodic Time to Update Local Clock Tone Latch

Description:

Define Periodic Time to Update Local Clock Tone Latch. Applies only when in flywheel mode. This define dicates the period at which the simulated 'last tone' time is updated. Units are seconds.

Limits

Not Applicable

Definition at line 509 of file cpu1_platform_cfg.h.

13.2.1.267 CFE_PLATFORM_TIME_CFG_SERVER

```
#define CFE_PLATFORM_TIME_CFG_SERVER true
```

Purpose Time Server or Time Client Selection

Description:

This configuration parameter selects whether the Time task functions as a time "server" or "client". A time server generates the "time at the tone" packet which is received by time clients.

Limits

Enable one, and only one by defining either CFE_PLATFORM_TIME_CFG_SERVER or CFE_PLATFORM_TIM↩E_CFG_CLIENT AS true. The other must be defined as false.

Definition at line 340 of file cpu1_platform_cfg.h.

13.2.1.268 CFE_PLATFORM_TIME_CFG_SIGNAL

```
#define CFE_PLATFORM_TIME_CFG_SIGNAL false
```

Purpose Include or Exclude the Primary/Redundant Tone Selection Cmd

Description:

Depending on the specific hardware system configuration, it may be possible to switch between a primary and redundant tone signal. If supported by hardware, this definitions will enable command interfaces to select the active tone signal. Both Time Clients and Time Servers support this feature. Note: Set the CFE_PLATFORM_TIME_C↩FG_SIGNAL define to true to enable tone signal commands.

Limits

Not Applicable

Definition at line 391 of file cpu1_platform_cfg.h.

13.2.1.269 CFE_PLATFORM_TIME_CFG_SOURCE

```
#define CFE_PLATFORM_TIME_CFG_SOURCE false
```

Purpose Include or Exclude the Internal/External Time Source Selection Cmd

Description:

By default, Time Servers maintain time using an internal MET which may be a h/w register or software counter, depending on available hardware. The following definition enables command interfaces to switch between an internal MET, or external time data received from one of several supported external time sources. Only a Time Server may be configured to use external time data. Note: Set the CFE_PLATFORM_TIME_CFG_SOURCE define to true to include the Time Source Selection Command (command allows selection between the internal or external time source). Then choose the external source with the CFE_TIME_CFG_SRC_??? define.

Limits

Only applies if CFE_PLATFORM_TIME_CFG_SERVER is set to true.

Definition at line 412 of file cpu1_platform_cfg.h.

13.2.1.270 CFE_PLATFORM_TIME_CFG_SRC_GPS

```
#define CFE_PLATFORM_TIME_CFG_SRC_GPS false
```

Definition at line 430 of file cpu1_platform_cfg.h.

13.2.1.271 CFE_PLATFORM_TIME_CFG_SRC_MET

```
#define CFE_PLATFORM_TIME_CFG_SRC_MET false
```

Purpose Choose the External Time Source for Server only

Description:

If CFE_PLATFORM_TIME_CFG_SOURCE is set to true, then one of the following external time source types must also be set to true. Do not set any of the external time source types to true unless CFE_PLATFORM_TIME_CF↩G_SOURCE is set to true.

Limits

1. If CFE_PLATFORM_TIME_CFG_SOURCE is set to true then one and only one of the following three external time sources can and must be set true: CFE_PLATFORM_TIME_CFG_SRC_MET, CFE_PLATFORM_TI↩ME_CFG_SRC_GPS, CFE_PLATFORM_TIME_CFG_SRC_TIME
2. Only applies if CFE_PLATFORM_TIME_CFG_SERVER is set to true.

Definition at line 429 of file cpu1_platform_cfg.h.

13.2.1.272 CFE_PLATFORM_TIME_CFG_SRC_TIME

```
#define CFE_PLATFORM_TIME_CFG_SRC_TIME false
```

Definition at line 431 of file cpu1_platform_cfg.h.

13.2.1.273 CFE_PLATFORM_TIME_CFG_START_FLY

```
#define CFE_PLATFORM_TIME_CFG_START_FLY 2
```

Purpose Define Time to Start Flywheel Since Last Tone

Description:

Define time to enter flywheel mode (in seconds since last tone data update) Units are microseconds as measured with the local clock.

Limits

Not Applicable

Definition at line 495 of file cpu1_platform_cfg.h.

13.2.1.274 CFE_PLATFORM_TIME_CFG_TONE_LIMIT

```
#define CFE_PLATFORM_TIME_CFG_TONE_LIMIT 20000
```

Purpose Define Timing Limits From One Tone To The Next

Description:

Defines limits to the timing of the 1Hz tone signal. A tone signal is valid only if it arrives within one second (plus or minus the tone limit) from the previous tone signal.Units are microseconds as measured with the local clock.

Limits

Not Applicable

Definition at line 481 of file cpu1_platform_cfg.h.

13.2.1.275 CFE_PLATFORM_TIME_CFG_VIRTUAL

```
#define CFE_PLATFORM_TIME_CFG_VIRTUAL true
```

Purpose Time Tone In Big-Endian Order

Description:

If this configuration parameter is defined, the CFE time server will publish time tones with payloads in big-endian order, and time clients will expect the tones to be in big-endian order. This is useful for mixed-endian environments. This will become obsolete once EDS is available and the CFE time tone message is defined.

Purpose Local MET or Virtual MET Selection for Time Servers

Description:

Depending on the specific hardware system configuration, it may be possible for Time Servers to read the "local" MET from a h/w register rather than having to track the MET as the count of tone signal interrupts (virtual MET)

Time Clients must be defined as using a virtual MET. Also, a Time Server cannot be defined as having both a h/w MET and an external time source (they both cannot synchronize to the same tone).

Note: "disable" this define (set to false) only for Time Servers with local hardware that supports a h/w MET that is synchronized to the tone signal !!!

Limits

Only applies if CFE_PLATFORM_TIME_CFG_SERVER is set to true.

Definition at line 375 of file cpu1_platform_cfg.h.

13.2.1.276 CFE_PLATFORM_TIME_MAX_DELTA_SECS

```
#define CFE_PLATFORM_TIME_MAX_DELTA_SECS 0
```

Purpose Define the Max Delta Limits for Time Servers using an Ext Time Source

Description:

If CFE_PLATFORM_TIME_CFG_SOURCE is set to true and one of the external time sources is also set to true, then the delta time limits for range checking is used.

When a new time value is received from an external source, the value is compared against the "expected" time value. If the delta exceeds the following defined amount, then the new time data will be ignored. This range checking is only performed after the clock state has been commanded to "valid". Until then, external time data is accepted unconditionally.

Limits

Applies only if both CFE_PLATFORM_TIME_CFG_SERVER and CFE_PLATFORM_TIME_CFG_SOURCE are set to true.

Definition at line 451 of file cpu1_platform_cfg.h.

13.2.1.277 CFE_PLATFORM_TIME_MAX_DELTA_SUBS

```
#define CFE_PLATFORM_TIME_MAX_DELTA_SUBS 500000
```

Definition at line 452 of file cpu1_platform_cfg.h.

13.2.1.278 CFE_PLATFORM_TIME_MAX_LOCAL_SECS

```
#define CFE_PLATFORM_TIME_MAX_LOCAL_SECS 27
```

Purpose Define the Local Clock Rollover Value in seconds and subseconds

Description:

Specifies the capability of the local clock. Indicates the time at which the local clock rolls over.

Limits

Not Applicable

Definition at line 465 of file cpu1_platform_cfg.h.

13.2.1.279 CFE_PLATFORM_TIME_MAX_LOCAL_SUBS

```
#define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS 0
```

Definition at line 466 of file cpu1_platform_cfg.h.

13.2.1.280 CFE_PLATFORM_TIME_START_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_START_TASK_PRIORITY 60
```

Purpose Define TIME Task Priorities

Description:

Defines the cFE_TIME Task priority. Defines the cFE_TIME Tone Task priority. Defines the cFE_TIME 1HZ Task priority.

Limits

There is a lower limit of zero and an upper limit of 255 on these configuration paramaters. Remember that the meaning of each task priority is inverted – a "lower" number has a "higher" priority.

Definition at line 1327 of file cpu1_platform_cfg.h.

13.2.1.281 CFE_PLATFORM_TIME_START_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

Purpose Define TIME Task Stack Sizes

Description:

Defines the cFE_TIME Main Task Stack Size Defines the cFE_TIME Tone Task Stack Size Defines the cFE_TIME 1HZ Task Stack Size

Limits

There is a lower limit of 2048 on these configuration paramaters. There are no restrictions on the upper limit however, the maximum stack size size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1346 of file cpu1_platform_cfg.h.

13.2.1.282 CFE_PLATFORM_TIME_TONE_TASK_PRIORITY

```
#define CFE_PLATFORM_TIME_TONE_TASK_PRIORITY 25
```

Definition at line 1328 of file cpu1_platform_cfg.h.

13.2.1.283 CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE

```
#define CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE 4096
```

Definition at line 1347 of file cpu1_platform_cfg.h.

13.2.1.284 CFE_SB_BUF_MEMORY_BYTES

```
#define CFE_SB_BUF_MEMORY_BYTES CFE_PLATFORM_SB_BUF_MEMORY_BYTES
```

Definition at line 1918 of file cpu1_platform_cfg.h.

13.2.1.285 CFE_SB_DEFAULT_MAP_FILENAME

```
#define CFE_SB_DEFAULT_MAP_FILENAME CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
```

Definition at line 1923 of file cpu1_platform_cfg.h.

13.2.1.286 CFE_SB_DEFAULT_MSG_LIMIT

```
#define CFE_SB_DEFAULT_MSG_LIMIT CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
```

Definition at line 1917 of file cpu1_platform_cfg.h.

13.2.1.287 CFE_SB_DEFAULT_PIPE_FILENAME

```
#define CFE_SB_DEFAULT_PIPE_FILENAME CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
```

Definition at line 1922 of file cpu1_platform_cfg.h.

13.2.1.288 CFE_SB_DEFAULT_REPORT_SENDER

```
#define CFE_SB_DEFAULT_REPORT_SENDER CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER
```

Definition at line 1957 of file cpu1_platform_cfg.h.

13.2.1.289 CFE_SB_DEFAULT_ROUTING_FILENAME

```
#define CFE_SB_DEFAULT_ROUTING_FILENAME CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME
```

Definition at line 1921 of file cpu1_platform_cfg.h.

13.2.1.290 CFE_SB_FILTER_MASK1

```
#define CFE_SB_FILTER_MASK1 CFE_PLATFORM_SB_FILTER_MASK1
```

Definition at line 1925 of file cpu1_platform_cfg.h.

13.2.1.291 CFE_SB_FILTER_MASK2

```
#define CFE_SB_FILTER_MASK2 CFE_PLATFORM_SB_FILTER_MASK2
```

Definition at line 1927 of file cpu1_platform_cfg.h.

13.2.1.292 CFE_SB_FILTER_MASK3

```
#define CFE_SB_FILTER_MASK3 CFE_PLATFORM_SB_FILTER_MASK3
```

Definition at line 1929 of file cpu1_platform_cfg.h.

13.2.1.293 CFE_SB_FILTER_MASK4

```
#define CFE_SB_FILTER_MASK4 CFE_PLATFORM_SB_FILTER_MASK4
```

Definition at line 1931 of file cpu1_platform_cfg.h.

13.2.1.294 CFE_SB_FILTER_MASK5

```
#define CFE_SB_FILTER_MASK5 CFE_PLATFORM_SB_FILTER_MASK5
```

Definition at line 1933 of file cpu1_platform_cfg.h.

13.2.1.295 CFE_SB_FILTER_MASK6

```
#define CFE_SB_FILTER_MASK6 CFE_PLATFORM_SB_FILTER_MASK6
```

Definition at line 1935 of file cpu1_platform_cfg.h.

13.2.1.296 CFE_SB_FILTER_MASK7

```
#define CFE_SB_FILTER_MASK7 CFE_PLATFORM_SB_FILTER_MASK7
```

Definition at line 1937 of file cpu1_platform_cfg.h.

13.2.1.297 CFE_SB_FILTER_MASK8

```
#define CFE_SB_FILTER_MASK8 CFE_PLATFORM_SB_FILTER_MASK8
```

Definition at line 1939 of file cpu1_platform_cfg.h.

13.2.1.298 CFE_SB_FILTERED_EVENT1

```
#define CFE_SB_FILTERED_EVENT1 CFE_PLATFORM_SB_FILTERED_EVENT1
```

Definition at line 1924 of file cpu1_platform_cfg.h.

13.2.1.299 CFE_SB_FILTERED_EVENT2

```
#define CFE_SB_FILTERED_EVENT2 CFE_PLATFORM_SB_FILTERED_EVENT2
```

Definition at line 1926 of file cpu1_platform_cfg.h.

13.2.1.300 CFE_SB_FILTERED_EVENT3

```
#define CFE_SB_FILTERED_EVENT3 CFE_PLATFORM_SB_FILTERED_EVENT3
```

Definition at line 1928 of file cpu1_platform_cfg.h.

13.2.1.301 CFE_SB_FILTERED_EVENT4

```
#define CFE_SB_FILTERED_EVENT4 CFE_PLATFORM_SB_FILTERED_EVENT4
```

Definition at line 1930 of file cpu1_platform_cfg.h.

13.2.1.302 CFE_SB_FILTERED_EVENT5

```
#define CFE_SB_FILTERED_EVENT5 CFE_PLATFORM_SB_FILTERED_EVENT5
```

Definition at line 1932 of file cpu1_platform_cfg.h.

13.2.1.303 CFE_SB_FILTERED_EVENT6

```
#define CFE_SB_FILTERED_EVENT6 CFE_PLATFORM_SB_FILTERED_EVENT6
```

Definition at line 1934 of file cpu1_platform_cfg.h.

13.2.1.304 CFE_SB_FILTERED_EVENT7

```
#define CFE_SB_FILTERED_EVENT7 CFE_PLATFORM_SB_FILTERED_EVENT7
```

Definition at line 1936 of file cpu1_platform_cfg.h.

13.2.1.305 CFE_SB_FILTERED_EVENT8

```
#define CFE_SB_FILTERED_EVENT8 CFE_PLATFORM_SB_FILTERED_EVENT8
```

Definition at line 1938 of file cpu1_platform_cfg.h.

13.2.1.306 CFE_SB_HIGHEST_VALID_MSGID

```
#define CFE_SB_HIGHEST_VALID_MSGID CFE_PLATFORM_SB_HIGHEST_VALID_MSGID
```

Definition at line 1920 of file cpu1_platform_cfg.h.

13.2.1.307 CFE_SB_MAX_BLOCK_SIZE

```
#define CFE_SB_MAX_BLOCK_SIZE CFE_PLATFORM_SB_MAX_BLOCK_SIZE
```

Definition at line 1956 of file cpu1_platform_cfg.h.

13.2.1.308 CFE_SB_MAX_DEST_PER_PKT

```
#define CFE_SB_MAX_DEST_PER_PKT CFE_PLATFORM_SB_MAX_DEST_PER_PKT
```

Definition at line 1916 of file cpu1_platform_cfg.h.

13.2.1.309 CFE_SB_MAX_MSG_IDS

```
#define CFE_SB_MAX_MSG_IDS CFE_PLATFORM_SB_MAX_MSG_IDS
```

Definition at line 1914 of file cpu1_platform_cfg.h.

13.2.1.310 CFE_SB_MAX_PIPE_DEPTH

```
#define CFE_SB_MAX_PIPE_DEPTH CFE_PLATFORM_SB_MAX_PIPE_DEPTH
```

Definition at line 1919 of file cpu1_platform_cfg.h.

13.2.1.311 CFE_SB_MAX_PIPES

```
#define CFE_SB_MAX_PIPES CFE_PLATFORM_SB_MAX_PIPES
```

Definition at line 1915 of file cpu1_platform_cfg.h.

13.2.1.312 CFE_SB_MEM_BLOCK_SIZE_01

```
#define CFE_SB_MEM_BLOCK_SIZE_01 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
```

Definition at line 1940 of file cpu1_platform_cfg.h.

13.2.1.313 CFE_SB_MEM_BLOCK_SIZE_02

```
#define CFE_SB_MEM_BLOCK_SIZE_02 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02
```

Definition at line 1941 of file cpu1_platform_cfg.h.

13.2.1.314 CFE_SB_MEM_BLOCK_SIZE_03

```
#define CFE_SB_MEM_BLOCK_SIZE_03 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03
```

Definition at line 1942 of file cpu1_platform_cfg.h.

13.2.1.315 CFE_SB_MEM_BLOCK_SIZE_04

```
#define CFE_SB_MEM_BLOCK_SIZE_04 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04
```

Definition at line 1943 of file cpu1_platform_cfg.h.

13.2.1.316 CFE_SB_MEM_BLOCK_SIZE_05

```
#define CFE_SB_MEM_BLOCK_SIZE_05 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05
```

Definition at line 1944 of file cpu1_platform_cfg.h.

13.2.1.317 CFE_SB_MEM_BLOCK_SIZE_06

```
#define CFE_SB_MEM_BLOCK_SIZE_06 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06
```

Definition at line 1945 of file cpu1_platform_cfg.h.

13.2.1.318 CFE_SB_MEM_BLOCK_SIZE_07

```
#define CFE_SB_MEM_BLOCK_SIZE_07 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07
```

Definition at line 1946 of file cpu1_platform_cfg.h.

13.2.1.319 CFE_SB_MEM_BLOCK_SIZE_08

```
#define CFE_SB_MEM_BLOCK_SIZE_08 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08
```

Definition at line 1947 of file cpu1_platform_cfg.h.

13.2.1.320 CFE_SB_MEM_BLOCK_SIZE_09

```
#define CFE_SB_MEM_BLOCK_SIZE_09 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09
```

Definition at line 1948 of file cpu1_platform_cfg.h.

13.2.1.321 CFE_SB_MEM_BLOCK_SIZE_10

```
#define CFE_SB_MEM_BLOCK_SIZE_10 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10
```

Definition at line 1949 of file cpu1_platform_cfg.h.

13.2.1.322 CFE_SB_MEM_BLOCK_SIZE_11

```
#define CFE_SB_MEM_BLOCK_SIZE_11 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11
```

Definition at line 1950 of file cpu1_platform_cfg.h.

13.2.1.323 CFE_SB_MEM_BLOCK_SIZE_12

```
#define CFE_SB_MEM_BLOCK_SIZE_12 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12
```

Definition at line 1951 of file cpu1_platform_cfg.h.

13.2.1.324 CFE_SB_MEM_BLOCK_SIZE_13

```
#define CFE_SB_MEM_BLOCK_SIZE_13 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13
```

Definition at line 1952 of file cpu1_platform_cfg.h.

13.2.1.325 CFE_SB_MEM_BLOCK_SIZE_14

```
#define CFE_SB_MEM_BLOCK_SIZE_14 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14
```

Definition at line 1953 of file cpu1_platform_cfg.h.

13.2.1.326 CFE_SB_MEM_BLOCK_SIZE_15

```
#define CFE_SB_MEM_BLOCK_SIZE_15 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15
```

Definition at line 1954 of file cpu1_platform_cfg.h.

13.2.1.327 CFE_SB_MEM_BLOCK_SIZE_16

```
#define CFE_SB_MEM_BLOCK_SIZE_16 CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16
```

Definition at line 1955 of file cpu1_platform_cfg.h.

13.2.1.328 CFE_SB_START_TASK_PRIORITY

```
#define CFE_SB_START_TASK_PRIORITY CFE_PLATFORM_SB_START_TASK_PRIORITY
```

Definition at line 2017 of file cpu1_platform_cfg.h.

13.2.1.329 CFE_SB_START_TASK_STACK_SIZE

```
#define CFE_SB_START_TASK_STACK_SIZE CFE_PLATFORM_SB_START_TASK_STACK_SIZE
```

Definition at line 2018 of file cpu1_platform_cfg.h.

13.2.1.330 CFE_TBL_BUF_MEMORY_BYTES

```
#define CFE_TBL_BUF_MEMORY_BYTES CFE_PLATFORM_TBL_BUF_MEMORY_BYTES
```

Definition at line 2074 of file cpu1_platform_cfg.h.

13.2.1.331 CFE_TBL_DEFAULT_REG_DUMP_FILE

```
#define CFE_TBL_DEFAULT_REG_DUMP_FILE CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE
```

Definition at line 2082 of file cpu1_platform_cfg.h.

13.2.1.332 CFE_TBL_MAX_CRITICAL_TABLES

```
#define CFE_TBL_MAX_CRITICAL_TABLES CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES
```

Definition at line 2078 of file cpu1_platform_cfg.h.

13.2.1.333 CFE_TBL_MAX_DBL_TABLE_SIZE

```
#define CFE_TBL_MAX_DBL_TABLE_SIZE CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE
```

Definition at line 2075 of file cpu1_platform_cfg.h.

13.2.1.334 CFE_TBL_MAX_NUM_HANDLES

```
#define CFE_TBL_MAX_NUM_HANDLES CFE_PLATFORM_TBL_MAX_NUM_HANDLES
```

Definition at line 2079 of file cpu1_platform_cfg.h.

13.2.1.335 CFE_TBL_MAX_NUM_TABLES

```
#define CFE_TBL_MAX_NUM_TABLES CFE_PLATFORM_TBL_MAX_NUM_TABLES
```

Definition at line 2077 of file cpu1_platform_cfg.h.

13.2.1.336 CFE_TBL_MAX_NUM_VALIDATIONS

```
#define CFE_TBL_MAX_NUM_VALIDATIONS CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS
```

Definition at line 2081 of file cpu1_platform_cfg.h.

13.2.1.337 CFE_TBL_MAX_SIMULTANEOUS_LOADS

```
#define CFE_TBL_MAX_SIMULTANEOUS_LOADS CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS
```

Definition at line 2080 of file cpu1_platform_cfg.h.

13.2.1.338 CFE_TBL_MAX_SNGL_TABLE_SIZE

```
#define CFE_TBL_MAX_SNGL_TABLE_SIZE CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE
```

Definition at line 2076 of file cpu1_platform_cfg.h.

13.2.1.339 CFE_TBL_START_TASK_PRIORITY

```
#define CFE_TBL_START_TASK_PRIORITY CFE_PLATFORM_TBL_START_TASK_PRIORITY
```

Definition at line 2027 of file cpu1_platform_cfg.h.

13.2.1.340 CFE_TBL_START_TASK_STACK_SIZE

```
#define CFE_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_TBL_START_TASK_STACK_SIZE
```

Definition at line 2028 of file cpu1_platform_cfg.h.

13.2.1.341 CFE_TBL_U32FROM4CHARS

```
#define CFE_TBL_U32FROM4CHARS CFE_PLATFORM_TBL_U32FROM4CHARS
```

Definition at line 2084 of file cpu1_platform_cfg.h.

13.2.1.342 CFE_TBL_VALID_PRID_1

```
#define CFE_TBL_VALID_PRID_1 CFE_PLATFORM_TBL_VALID_PRID_1
```

Definition at line 2088 of file cpu1_platform_cfg.h.

13.2.1.343 CFE_TBL_VALID_PRID_2

```
#define CFE_TBL_VALID_PRID_2 CFE_PLATFORM_TBL_VALID_PRID_2
```

Definition at line 2089 of file cpu1_platform_cfg.h.

13.2.1.344 CFE_TBL_VALID_PRID_3

```
#define CFE_TBL_VALID_PRID_3 CFE_PLATFORM_TBL_VALID_PRID_3
```

Definition at line 2090 of file cpu1_platform_cfg.h.

13.2.1.345 CFE_TBL_VALID_PRID_4

```
#define CFE_TBL_VALID_PRID_4 CFE_PLATFORM_TBL_VALID_PRID_4
```

Definition at line 2091 of file cpu1_platform_cfg.h.

13.2.1.346 CFE_TBL_VALID_PRID_COUNT

```
#define CFE_TBL_VALID_PRID_COUNT CFE_PLATFORM_TBL_VALID_PRID_COUNT
```

Definition at line 2087 of file cpu1_platform_cfg.h.

13.2.1.347 CFE_TBL_VALID_SCID_1

```
#define CFE_TBL_VALID_SCID_1 CFE_PLATFORM_TBL_VALID_SCID_1
```

Definition at line 2085 of file cpu1_platform_cfg.h.

13.2.1.348 CFE_TBL_VALID_SCID_2

```
#define CFE_TBL_VALID_SCID_2 CFE_PLATFORM_TBL_VALID_SCID_2
```

Definition at line 2086 of file cpu1_platform_cfg.h.

13.2.1.349 CFE_TBL_VALID_SCID_COUNT

```
#define CFE_TBL_VALID_SCID_COUNT CFE_PLATFORM_TBL_VALID_SCID_COUNT
```

Definition at line 2083 of file cpu1_platform_cfg.h.

13.2.1.350 CFE_TIME_1HZ_TASK_PRIORITY

```
#define CFE_TIME_1HZ_TASK_PRIORITY CFE_PLATFORM_TIME_1HZ_TASK_PRIORITY
```

Definition at line 2023 of file cpu1_platform_cfg.h.

13.2.1.351 CFE_TIME_1HZ_TASK_STACK_SIZE

```
#define CFE_TIME_1HZ_TASK_STACK_SIZE CFE_PLATFORM_TIME_1HZ_TASK_STACK_SIZE
```

Definition at line 2026 of file cpu1_platform_cfg.h.

13.2.1.352 CFE_TIME_CFG_CLIENT

```
#define CFE_TIME_CFG_CLIENT CFE_PLATFORM_TIME_CFG_CLIENT
```

Definition at line 1959 of file cpu1_platform_cfg.h.

13.2.1.353 CFE_TIME_CFG_LATCH_FLY

```
#define CFE_TIME_CFG_LATCH_FLY CFE_PLATFORM_TIME_CFG_LATCH_FLY
```

Definition at line 1972 of file cpu1_platform_cfg.h.

13.2.1.354 CFE_TIME_CFG_SERVER

```
#define CFE_TIME_CFG_SERVER CFE_PLATFORM_TIME_CFG_SERVER
```

Definition at line 1958 of file cpu1_platform_cfg.h.

13.2.1.355 CFE_TIME_CFG_SIGNAL

```
#define CFE_TIME_CFG_SIGNAL CFE_PLATFORM_TIME_CFG_SIGNAL
```

Definition at line 1961 of file cpu1_platform_cfg.h.

13.2.1.356 CFE_TIME_CFG_SOURCE

```
#define CFE_TIME_CFG_SOURCE CFE_PLATFORM_TIME_CFG_SOURCE
```

Definition at line 1962 of file cpu1_platform_cfg.h.

13.2.1.357 CFE_TIME_CFG_SRC_GPS

```
#define CFE_TIME_CFG_SRC_GPS CFE_PLATFORM_TIME_CFG_SRC_GPS
```

Definition at line 1964 of file cpu1_platform_cfg.h.

13.2.1.358 CFE_TIME_CFG_SRC_MET

```
#define CFE_TIME_CFG_SRC_MET CFE_PLATFORM_TIME_CFG_SRC_MET
```

Definition at line 1963 of file cpu1_platform_cfg.h.

13.2.1.359 CFE_TIME_CFG_SRC_TIME

```
#define CFE_TIME_CFG_SRC_TIME CFE_PLATFORM_TIME_CFG_SRC_TIME
```

Definition at line 1965 of file cpu1_platform_cfg.h.

13.2.1.360 CFE_TIME_CFG_START_FLY

```
#define CFE_TIME_CFG_START_FLY CFE_PLATFORM_TIME_CFG_START_FLY
```

Definition at line 1971 of file cpu1_platform_cfg.h.

13.2.1.361 CFE_TIME_CFG_TONE_LIMIT

```
#define CFE_TIME_CFG_TONE_LIMIT CFE_PLATFORM_TIME_CFG_TONE_LIMIT
```

Definition at line 1970 of file cpu1_platform_cfg.h.

13.2.1.362 CFE_TIME_CFG_VIRTUAL

```
#define CFE_TIME_CFG_VIRTUAL CFE_PLATFORM_TIME_CFG_VIRTUAL
```

Definition at line 1960 of file cpu1_platform_cfg.h.

13.2.1.363 CFE_TIME_ENA_1HZ_CMD_PKT

```
#define CFE_TIME_ENA_1HZ_CMD_PKT true
```

Definition at line 2101 of file cpu1_platform_cfg.h.

13.2.1.364 CFE_TIME_MAX_DELTA_SECS

```
#define CFE_TIME_MAX_DELTA_SECS CFE_PLATFORM_TIME_MAX_DELTA_SECS
```

Definition at line 1966 of file cpu1_platform_cfg.h.

13.2.1.365 CFE_TIME_MAX_DELTA_SUBS

```
#define CFE_TIME_MAX_DELTA_SUBS CFE_PLATFORM_TIME_MAX_DELTA_SUBS
```

Definition at line 1967 of file cpu1_platform_cfg.h.

13.2.1.366 CFE_TIME_MAX_LOCAL_SECS

```
#define CFE_TIME_MAX_LOCAL_SECS CFE_PLATFORM_TIME_MAX_LOCAL_SECS
```

Definition at line 1968 of file cpu1_platform_cfg.h.

13.2.1.367 CFE_TIME_MAX_LOCAL_SUBS

```
#define CFE_TIME_MAX_LOCAL_SUBS CFE_PLATFORM_TIME_MAX_LOCAL_SUBS
```

Definition at line 1969 of file cpu1_platform_cfg.h.

13.2.1.368 CFE_TIME_START_TASK_PRIORITY

```
#define CFE_TIME_START_TASK_PRIORITY CFE_PLATFORM_TIME_START_TASK_PRIORITY
```

Definition at line 2021 of file cpu1_platform_cfg.h.

13.2.1.369 CFE_TIME_START_TASK_STACK_SIZE

```
#define CFE_TIME_START_TASK_STACK_SIZE CFE_PLATFORM_TIME_START_TASK_STACK_SIZE
```

Definition at line 2024 of file cpu1_platform_cfg.h.

13.2.1.370 CFE_TIME_TONE_TASK_PRIORITY

```
#define CFE_TIME_TONE_TASK_PRIORITY CFE_PLATFORM_TIME_TONE_TASK_PRIORITY
```

Definition at line 2022 of file cpu1_platform_cfg.h.

13.2.1.371 CFE_TIME_TONE_TASK_STACK_SIZE

```
#define CFE_TIME_TONE_TASK_STACK_SIZE CFE_PLATFORM_TIME_TONE_TASK_STACK_SIZE
```

Definition at line 2025 of file cpu1_platform_cfg.h.

13.3 default_osconfig.h File Reference

Macros

- #define OS_MAX_TASKS 64
- #define OS_MAX_QUEUES 64
- #define OS_MAX_COUNT_SEMAPHORES 20
- #define OS_MAX_BIN_SEMAPHORES 20
- #define OS_MAX_MUTEXES 20
- #define OS_MAX_PATH_LEN 64
- #define OS_MAX_LOCAL_PATH_LEN (OS_MAX_PATH_LEN + OS_FS_PHYS_NAME_LEN)
- #define OS_MAX_API_NAME 20
- #define OS_MAX_FILE_NAME 20
- #define OS_BUFFER_SIZE 172
- #define OS_BUFFER_MSG_DEPTH 100
- #define OS_UTILITY_TASK_ON
- #define OS_UTILITYTASK_STACK_SIZE 2048
- #define OS_UTILITYTASK_PRIORITY 245
- #define OS_MAX_CMD_LEN 1000
- #define OS_INCLUDE_NETWORK
- #define OS_MAX_NUM_OPEN_FILES 50
- #define OS_SHELL_CMD_INPUT_FILE_NAME "/ram/OS_ShellCmd.in"
- #define OS_INCLUDE_MODULE_LOADER
- #define OS_MAX_MODULES 32
- #define OS_MAX_SYM_LEN 64
- #define OS_MAX_TIMEBASES 5
- #define OS_MAX_TIMERS 5
- #define OS_MAX_NUM_OPEN_DIRS 4

13.3.1 Macro Definition Documentation

13.3.1.1 OS_BUFFER_MSG_DEPTH

```
#define OS_BUFFER_MSG_DEPTH 100
```

Definition at line 72 of file default_osconfig.h.

13.3.1.2 OS_BUFFER_SIZE

```
#define OS_BUFFER_SIZE 172
```

Definition at line 71 of file default_osconfig.h.

13.3.1.3 OS_INCLUDE_MODULE_LOADER

```
#define OS_INCLUDE_MODULE_LOADER
```

Definition at line 125 of file default_osconfig.h.

13.3.1.4 OS_INCLUDE_NETWORK

```
#define OS_INCLUDE_NETWORK
```

Definition at line 103 of file default_osconfig.h.

13.3.1.5 OS_MAX_API_NAME

```
#define OS_MAX_API_NAME 20
```

Definition at line 61 of file default_osconfig.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteCDS(), CFE_ES_FormCDSName(), CFE_ES_GetAppIDByName(), CFE_ES_GetGenCounterIDByName(), CFE_ES_Get↩ TaskInfo(), CFE_ES_ListApplications(), CFE_ES_ListTasks(), CFE_ES_PoolCreateEx(), CFE_ES_QueryOneCmd(), CFE_ES_RegisterCDS(), CFE_ES_RegisterGenCounter(), CFE_ES_ReloadAppCmd(), CFE_ES_RestartAppCmd(), CFE_ES_StartAppCmd(), CFE_ES_StopAppCmd(), CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilter↩ Cmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEvents↩ Cmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_WriteAppDataFileCmd(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetAppTskName(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_Send↩ MsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), EVS_IsFiltered(), and E↩ VS_NotRegistered().

13.3.1.6 OS_MAX_BIN_SEMAPHORES

```
#define OS_MAX_BIN_SEMAPHORES 20
```

Definition at line 43 of file default_osconfig.h.

13.3.1.7 OS_MAX_CMD_LEN

```
#define OS_MAX_CMD_LEN 1000
```

Definition at line 96 of file default_osconfig.h.

13.3.1.8 OS_MAX_COUNT_SEMAPHORES

```
#define OS_MAX_COUNT_SEMAPHORES 20
```

Definition at line 42 of file default_osconfig.h.

13.3.1.9 OS_MAX_FILE_NAME

```
#define OS_MAX_FILE_NAME 20
```

Definition at line 66 of file default_osconfig.h.

13.3.1.10 OS_MAX_LOCAL_PATH_LEN

```
#define OS_MAX_LOCAL_PATH_LEN (OS_MAX_PATH_LEN + OS_FS_PHYS_NAME_LEN)
```

Definition at line 56 of file default_osconfig.h.

13.3.1.11 OS_MAX_MODULES

```
#define OS_MAX_MODULES 32
```

Definition at line 134 of file default_osconfig.h.

13.3.1.12 OS_MAX_MUTEXES

```
#define OS_MAX_MUTEXES 20
```

Definition at line 44 of file default_osconfig.h.

13.3.1.13 OS_MAX_NUM_OPEN_DIRS

```
#define OS_MAX_NUM_OPEN_DIRS 4
```

Definition at line 174 of file default_osconfig.h.

13.3.1.14 OS_MAX_NUM_OPEN_FILES

```
#define OS_MAX_NUM_OPEN_FILES 50
```

Definition at line 108 of file default_osconfig.h.

13.3.1.15 OS_MAX_PATH_LEN

```
#define OS_MAX_PATH_LEN 64
```

Definition at line 49 of file default_osconfig.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_LoadLibrary(), CFE_ES_Query↩AllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_ReloadApp(), CFE_ES_ReloadAppCmd(), CFE_ES_ShellCmd(), CFE_ES_StartAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_WriteERLogCmd(), CFE_ES_WriteSyslogCmd(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfoCmd(), CFE_SB_↩SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

13.3.1.16 OS_MAX_QUEUES

```
#define OS_MAX_QUEUES 64
```

Definition at line 41 of file default_osconfig.h.

13.3.1.17 OS_MAX_SYM_LEN

```
#define OS_MAX_SYM_LEN 64
```

Definition at line 148 of file default_osconfig.h.

13.3.1.18 OS_MAX_TASKS

```
#define OS_MAX_TASKS 64
```

Definition at line 40 of file default_osconfig.h.

Referenced by CFE_ES_CleanUpApp(), CFE_ES_DeleteChildTask(), CFE_ES_GetAppInfoInternal(), CFE_ES_Get↩TaskInfo(), CFE_ES_ListTasks(), CFE_ES_Main(), CFE_ES_ProcessCoreException(), and CFE_ES_QueryAllTasks↩Cmd().

13.3.1.19 OS_MAX_TIMEBASES

```
#define OS_MAX_TIMEBASES 5
```

Definition at line 157 of file default_osconfig.h.

13.3.1.20 OS_MAX_TIMERS

```
#define OS_MAX_TIMERS 5
```

Definition at line 168 of file default_osconfig.h.

13.3.1.21 OS_SHELL_CMD_INPUT_FILE_NAME

```
#define OS_SHELL_CMD_INPUT_FILE_NAME "/ram/OS_ShellCmd.in"
```

Definition at line 114 of file default_osconfig.h.

13.3.1.22 OS_UTILITY_TASK_ON

```
#define OS_UTILITY_TASK_ON
```

Definition at line 83 of file default_osconfig.h.

13.3.1.23 OS_UTILITYTASK_PRIORITY

```
#define OS_UTILITYTASK_PRIORITY 245
```

Definition at line 89 of file default_osconfig.h.

13.3.1.24 OS_UTILITYTASK_STACK_SIZE

```
#define OS_UTILITYTASK_STACK_SIZE 2048
```

Definition at line 87 of file default_osconfig.h.

13.4 sample_mission_cfg.h File Reference

Macros

- #define CFE_MISSION_SPACECRAFT_ID 0x42
- #define MESSAGE_FORMAT_IS_CCSDS
- #define CFE_MISSION_SB_PACKET_TIME_FORMAT CFE_MISSION_SB_TIME_32_16_SUBS
- #define CFE_MISSION_SB_MAX_SB_MSG_SIZE 32768
- #define CFE_MISSION_TIME_CFG_DEFAULT_TAI true
- #define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
- #define CFE_MISSION_TIME_CFG_FAKE_TONE true
- #define CFE_MISSION_TIME_AT_TONE_WAS true
- #define CFE_MISSION_TIME_AT_TONE_WILL_BE false
- #define CFE_MISSION_TIME_MIN_ELAPSED 0
- #define CFE_MISSION_TIME_MAX_ELAPSED 200000
- #define CFE_MISSION_TIME_DEF_MET_SECS 1000
- #define CFE_MISSION_TIME_DEF_MET_SUBS 0
- #define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
- #define CFE_MISSION_TIME_DEF_STCF_SUBS 0
- #define CFE_MISSION_TIME_DEF_LEAPS 32
- #define CFE_MISSION_TIME_DEF_DELAY_SECS 0
- #define CFE_MISSION_TIME_DEF_DELAY_SUBS 1000
- #define CFE_MISSION_TIME_EPOCH_YEAR 1980
- #define CFE_MISSION_TIME_EPOCH_DAY 1
- #define CFE_MISSION_TIME_EPOCH_HOUR 0
- #define CFE_MISSION_TIME_EPOCH_MINUTE 0
- #define CFE_MISSION_TIME_EPOCH_SECOND 0
- #define CFE_MISSION_TIME_FS_FACTOR 789004800
- #define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH 16
- #define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122
- #define CFE_MISSION_ES_DEFAULT_CRC CFE_MISSION_ES_CRC_16
- #define CFE_MISSION_TBL_MAX_NAME_LENGTH 16
- #define CFE_MISSION_CMD_MID_BASE1 0x1800
- #define CFE_MISSION_TLM_MID_BASE1 0x0800
- #define CFE_MISSION_CMD_APPID_BASE1 1
- #define CFE_MISSION_TLM_APPID_BASE1 0
- #define CFE_MISSION_CMD_MID_BASE_GLOB 0x1860
- #define CFE_MISSION_TLM_MID_BASE_GLOB 0x0860
- #define CFE_MISSION_EVS_CMD_MSG 1
- #define CFE_MISSION_SB_CMD_MSG 3
- #define CFE_MISSION_TBL_CMD_MSG 4
- #define CFE_MISSION_TIME_CMD_MSG 5
- #define CFE_MISSION_ES_CMD_MSG 6

- #define CFE_MISSION_ES_SEND_HK_MSG 8
- #define CFE_MISSION_EVS_SEND_HK_MSG 9
- #define CFE_MISSION_SB_SEND_HK_MSG 11
- #define CFE_MISSION_TBL_SEND_HK_MSG 12
- #define CFE_MISSION_TIME_SEND_HK_MSG 13
- #define CFE_MISSION_TIME_TONE_CMD_MSG 16
- #define CFE_MISSION_TIME_1HZ_CMD_MSG 17
- #define CFE_MISSION_TIME_DATA_CMD_MSG 0
- #define CFE_MISSION_TIME_SEND_CMD_MSG 2
- #define CFE_MISSION_ES_HK_TLM_MSG 0
- #define CFE_MISSION_EVS_HK_TLM_MSG 1
- #define CFE_MISSION_SB_HK_TLM_MSG 3
- #define CFE_MISSION_TBL_HK_TLM_MSG 4
- #define CFE_MISSION_TIME_HK_TLM_MSG 5
- #define CFE_MISSION_TIME_DIAG_TLM_MSG 6
- #define CFE_MISSION_EVS_LONG_EVENT_MSG_MSG 8
- #define CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG 9
- #define CFE_MISSION_SB_STATS_TLM_MSG 10
- #define CFE_MISSION_ES_APP_TLM_MSG 11
- #define CFE_MISSION_TBL_REG_TLM_MSG 12
- #define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13
- #define CFE_MISSION_SB_ONESUB_TLM_MSG 14
- #define CFE_MISSION_ES_SHELL_TLM_MSG 15
- #define CFE_MISSION_ES_MEMSTATS_TLM_MSG 16
- #define CFE_MISSION_ES_MAX_APPLICATIONS 16
- #define CFE_MISSION_ES_MAX_SHELL_CMD 64
- #define CFE_MISSION_ES_MAX_SHELL_PKT 64
- #define CFE_MISSION_ES_PERF_MAX_IDS 128
- #define CFE_MISSION_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_↩ MISSION_MAX_API_LEN + 4)
- #define CFE_MISSION_SB_MAX_PIPES 64
- #define CFE_MISSION_MAX_PATH_LEN 64
- #define CFE_MISSION_MAX_FILE_LEN 20
- #define CFE_MISSION_MAX_API_LEN 20
- #define CFE_MISSION_ES_CDS_MAX_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + CF↩ E_MISSION_MAX_API_LEN + 4)
- #define CFE_SPACECRAFT_ID CFE_MISSION_SPACECRAFT_ID
- #define CFE_SB_TIME_32_16_SUBS CFE_MISSION_SB_TIME_32_16_SUBS
- #define CFE_SB_TIME_32_32_SUBS CFE_MISSION_SB_TIME_32_32_SUBS
- #define CFE_SB_TIME_32_32_M_20 CFE_MISSION_SB_TIME_32_32_M_20
- #define CFE_SB_PACKET_TIME_FORMAT CFE_MISSION_SB_PACKET_TIME_FORMAT
- #define CFE_SB_MAX_SB_MSG_SIZE CFE_MISSION_SB_MAX_SB_MSG_SIZE
- #define CFE_TIME_CFG_DEFAULT_TAI CFE_MISSION_TIME_CFG_DEFAULT_TAI
- #define CFE_TIME_CFG_DEFAULT_UTC CFE_MISSION_TIME_CFG_DEFAULT_UTC
- #define CFE_TIME_CFG_FAKE_TONE CFE_MISSION_TIME_CFG_FAKE_TONE
- #define CFE_TIME_AT_TONE_WAS CFE_MISSION_TIME_AT_TONE_WAS
- #define CFE_TIME_AT_TONE_WILL_BE CFE_MISSION_TIME_AT_TONE_WILL_BE
- #define CFE_TIME_MIN_ELAPSED CFE_MISSION_TIME_MIN_ELAPSED
- #define CFE_TIME_MAX_ELAPSED CFE_MISSION_TIME_MAX_ELAPSED
- #define CFE_TIME_DEF_MET_SECS CFE_MISSION_TIME_DEF_MET_SECS
- #define CFE_TIME_DEF_MET_SUBS CFE_MISSION_TIME_DEF_MET_SUBS

- #define CFE_TIME_DEF_STCF_SECS CFE_MISSION_TIME_DEF_STCF_SECS
- #define CFE_TIME_DEF_STCF_SUBS CFE_MISSION_TIME_DEF_STCF_SUBS
- #define CFE_TIME_DEF_LEAPS CFE_MISSION_TIME_DEF_LEAPS
- #define CFE_TIME_DEF_DELAY_SECS CFE_MISSION_TIME_DEF_DELAY_SECS
- #define CFE_TIME_DEF_DELAY_SUBS CFE_MISSION_TIME_DEF_DELAY_SUBS
- #define CFE_TIME_EPOCH_YEAR CFE_MISSION_TIME_EPOCH_YEAR
- #define CFE_TIME_EPOCH_DAY CFE_MISSION_TIME_EPOCH_DAY
- #define CFE_TIME_EPOCH_HOUR CFE_MISSION_TIME_EPOCH_HOUR
- #define CFE_TIME_EPOCH_MINUTE CFE_MISSION_TIME_EPOCH_MINUTE
- #define CFE_TIME_EPOCH_SECOND CFE_MISSION_TIME_EPOCH_SECOND
- #define CFE_TIME_FS_FACTOR CFE_MISSION_TIME_FS_FACTOR
- #define CFE_ES_CDS_MAX_NAME_LENGTH CFE_MISSION_ES_CDS_MAX_NAME_LENGTH
- #define CFE_EVS_MAX_MESSAGE_LENGTH CFE_MISSION_EVS_MAX_MESSAGE_LENGTH
- #define CFE_ES_CRC_8 CFE_MISSION_ES_CRC_8
- #define CFE_ES_CRC_16 CFE_MISSION_ES_CRC_16
- #define CFE_ES_CRC_32 CFE_MISSION_ES_CRC_32
- #define CFE_ES_DEFAULT_CRC CFE_MISSION_ES_DEFAULT_CRC
- #define CFE_TBL_MAX_NAME_LENGTH CFE_MISSION_TBL_MAX_NAME_LENGTH
- #define CFE_CMD_MID_BASE_CPU1 CFE_MISSION_CMD_MID_BASE_CPU1
- #define CFE_TLM_MID_BASE_CPU1 CFE_MISSION_TLM_MID_BASE_CPU1
- #define CFE_CMD_APPID_BASE_CPU1 CFE_MISSION_CMD_APPID_BASE_CPU1
- #define CFE_TLM_APPID_BASE_CPU1 CFE_MISSION_TLM_APPID_BASE_CPU1
- #define CFE_CMD_MID_BASE_CPU2 CFE_MISSION_CMD_MID_BASE_CPU2
- #define CFE_TLM_MID_BASE_CPU2 CFE_MISSION_TLM_MID_BASE_CPU2
- #define CFE_CMD_APPID_BASE_CPU2 CFE_MISSION_CMD_APPID_BASE_CPU2
- #define CFE_TLM_APPID_BASE_CPU2 CFE_MISSION_TLM_APPID_BASE_CPU2
- #define CFE_CMD_MID_BASE_CPU3 CFE_MISSION_CMD_MID_BASE_CPU3
- #define CFE_TLM_MID_BASE_CPU3 CFE_MISSION_TLM_MID_BASE_CPU3
- #define CFE_CMD_APPID_BASE_CPU3 CFE_MISSION_CMD_APPID_BASE_CPU3
- #define CFE_TLM_APPID_BASE_CPU3 CFE_MISSION_TLM_APPID_BASE_CPU3
- #define CFE_CMD_MID_BASE_GLOB CFE_MISSION_CMD_MID_BASE_GLOB
- #define CFE_TLM_MID_BASE_GLOB CFE_MISSION_TLM_MID_BASE_GLOB
- #define CFE_EVS_CMD_MSG CFE_MISSION_EVS_CMD_MSG
- #define CFE_SB_CMD_MSG CFE_MISSION_SB_CMD_MSG
- #define CFE_TBL_CMD_MSG CFE_MISSION_TBL_CMD_MSG
- #define CFE_TIME_CMD_MSG CFE_MISSION_TIME_CMD_MSG
- #define CFE_ES_CMD_MSG CFE_MISSION_ES_CMD_MSG
- #define CFE_ES_SEND_HK_MSG CFE_MISSION_ES_SEND_HK_MSG
- #define CFE_EVS_SEND_HK_MSG CFE_MISSION_EVS_SEND_HK_MSG
- #define CFE_SB_SEND_HK_MSG CFE_MISSION_SB_SEND_HK_MSG
- #define CFE_TBL_SEND_HK_MSG CFE_MISSION_TBL_SEND_HK_MSG
- #define CFE_TIME_SEND_HK_MSG CFE_MISSION_TIME_SEND_HK_MSG
- #define CFE_TIME_TONE_CMD_MSG CFE_MISSION_TIME_TONE_CMD_MSG
- #define CFE_TIME_1HZ_CMD_MSG CFE_MISSION_TIME_1HZ_CMD_MSG
- #define CFE_TIME_DATA_CMD_MSG CFE_MISSION_TIME_DATA_CMD_MSG
- #define CFE_TIME_SEND_CMD_MSG CFE_MISSION_TIME_SEND_CMD_MSG
- #define CFE_ES_HK_TLM_MSG CFE_MISSION_ES_HK_TLM_MSG
- #define CFE_EVS_HK_TLM_MSG CFE_MISSION_EVS_HK_TLM_MSG
- #define CFE_SB_HK_TLM_MSG CFE_MISSION_SB_HK_TLM_MSG
- #define CFE_TBL_HK_TLM_MSG CFE_MISSION_TBL_HK_TLM_MSG
- #define CFE_TIME_HK_TLM_MSG CFE_MISSION_TIME_HK_TLM_MSG

- #define CFE_TIME_DIAG_TLM_MSG CFE_MISSION_TIME_DIAG_TLM_MSG
- #define CFE_EVS_EVENT_MSG_MSG CFE_MISSION_EVS_LONG_EVENT_MSG_MSG
- #define CFE_SB_STATS_TLM_MSG CFE_MISSION_SB_STATS_TLM_MSG
- #define CFE_ES_APP_TLM_MSG CFE_MISSION_ES_APP_TLM_MSG
- #define CFE_TBL_REG_TLM_MSG CFE_MISSION_TBL_REG_TLM_MSG
- #define CFE_SB_ALLSUBS_TLM_MSG CFE_MISSION_SB_ALLSUBS_TLM_MSG
- #define CFE_SB_ONESUB_TLM_MSG CFE_MISSION_SB_ONESUB_TLM_MSG
- #define CFE_ES_SHELL_TLM_MSG CFE_MISSION_ES_SHELL_TLM_MSG
- #define CFE_ES_MEMSTATS_TLM_MSG CFE_MISSION_ES_MEMSTATS_TLM_MSG

Packet timestamp format identifiers

- #define CFE_MISSION_SB_TIME_32_16_SUBS 1
  32 bits seconds + 16 bits subseconds (units = 2^^-16)
- #define CFE_MISSION_SB_TIME_32_32_SUBS 2
  32 bits seconds + 32 bits subseconds (units = 2^^-32)
- #define CFE_MISSION_SB_TIME_32_32_M_20 3
  32 bits seconds + 20 bits microsecs + 12 bits reserved

Checksum/CRC algorithm identifiers

- #define CFE_MISSION_ES_CRC_8 1
  CRC (8 bit additive - returns 32 bit total) (Currently not implemented)
- #define CFE_MISSION_ES_CRC_16 2
  CRC (16 bit additive - returns 32 bit total)
- #define CFE_MISSION_ES_CRC_32 3
  CRC (32 bit additive - returns 32 bit total) (Currently not implemented)

13.4.1 Macro Definition Documentation

13.4.1.1 CFE_CMD_APPID_BASE_CPU1

```
#define CFE_CMD_APPID_BASE_CPU1 CFE_MISSION_CMD_APPID_BASE_CPU1
```

Definition at line 736 of file sample_mission_cfg.h.

13.4.1.2 CFE_CMD_APPID_BASE_CPU2

```
#define CFE_CMD_APPID_BASE_CPU2 CFE_MISSION_CMD_APPID_BASE_CPU2
```

Definition at line 740 of file sample_mission_cfg.h.

13.4.1.3 CFE_CMD_APPID_BASE_CPU3

```
#define CFE_CMD_APPID_BASE_CPU3 CFE_MISSION_CMD_APPID_BASE_CPU3
```

Definition at line 744 of file sample_mission_cfg.h.

13.4.1.4 CFE_CMD_MID_BASE_CPU1

```
#define CFE_CMD_MID_BASE_CPU1 CFE_MISSION_CMD_MID_BASE_CPU1
```

Definition at line 734 of file sample_mission_cfg.h.

13.4.1.5 CFE_CMD_MID_BASE_CPU2

```
#define CFE_CMD_MID_BASE_CPU2 CFE_MISSION_CMD_MID_BASE_CPU2
```

Definition at line 738 of file sample_mission_cfg.h.

13.4.1.6 CFE_CMD_MID_BASE_CPU3

```
#define CFE_CMD_MID_BASE_CPU3 CFE_MISSION_CMD_MID_BASE_CPU3
```

Definition at line 742 of file sample_mission_cfg.h.

13.4.1.7 CFE_CMD_MID_BASE_GLOB

```
#define CFE_CMD_MID_BASE_GLOB CFE_MISSION_CMD_MID_BASE_GLOB
```

Definition at line 746 of file sample_mission_cfg.h.

13.4.1.8 CFE_ES_APP_TLM_MSG

```
#define CFE_ES_APP_TLM_MSG CFE_MISSION_ES_APP_TLM_MSG
```

Definition at line 770 of file sample_mission_cfg.h.

13.4.1.9 CFE_ES_CDS_MAX_NAME_LENGTH

```
#define CFE_ES_CDS_MAX_NAME_LENGTH CFE_MISSION_ES_CDS_MAX_NAME_LENGTH
```

Definition at line 727 of file sample_mission_cfg.h.

13.4.1.10 CFE_ES_CMD_MSG

```
#define CFE_ES_CMD_MSG CFE_MISSION_ES_CMD_MSG
```

Definition at line 752 of file sample_mission_cfg.h.

13.4.1.11 CFE_ES_CRC_16

```
#define CFE_ES_CRC_16 CFE_MISSION_ES_CRC_16
```

Definition at line 730 of file sample_mission_cfg.h.

13.4.1.12 CFE_ES_CRC_32

```
#define CFE_ES_CRC_32 CFE_MISSION_ES_CRC_32
```

Definition at line 731 of file sample_mission_cfg.h.

13.4.1.13 CFE_ES_CRC_8

```
#define CFE_ES_CRC_8 CFE_MISSION_ES_CRC_8
```

Definition at line 729 of file sample_mission_cfg.h.

13.4.1.14 CFE_ES_DEFAULT_CRC

```
#define CFE_ES_DEFAULT_CRC CFE_MISSION_ES_DEFAULT_CRC
```

Definition at line 732 of file sample_mission_cfg.h.

13.4.1.15 CFE_ES_HK_TLM_MSG

```
#define CFE_ES_HK_TLM_MSG CFE_MISSION_ES_HK_TLM_MSG
```

Definition at line 762 of file sample_mission_cfg.h.

13.4.1.16 CFE_ES_MEMSTATS_TLM_MSG

```
#define CFE_ES_MEMSTATS_TLM_MSG CFE_MISSION_ES_MEMSTATS_TLM_MSG
```

Definition at line 775 of file sample_mission_cfg.h.

13.4.1.17 CFE_ES_SEND_HK_MSG

```
#define CFE_ES_SEND_HK_MSG CFE_MISSION_ES_SEND_HK_MSG
```

Definition at line 753 of file sample_mission_cfg.h.

13.4.1.18 CFE_ES_SHELL_TLM_MSG

```
#define CFE_ES_SHELL_TLM_MSG CFE_MISSION_ES_SHELL_TLM_MSG
```

Definition at line 774 of file sample_mission_cfg.h.

13.4.1.19 CFE_EVS_CMD_MSG

```
#define CFE_EVS_CMD_MSG CFE_MISSION_EVS_CMD_MSG
```

Definition at line 748 of file sample_mission_cfg.h.

13.4.1.20 CFE_EVS_EVENT_MSG_MSG

```
#define CFE_EVS_EVENT_MSG_MSG CFE_MISSION_EVS_LONG_EVENT_MSG_MSG
```

Definition at line 768 of file sample_mission_cfg.h.

13.4.1.21 CFE_EVS_HK_TLM_MSG

```
#define CFE_EVS_HK_TLM_MSG CFE_MISSION_EVS_HK_TLM_MSG
```

Definition at line 763 of file sample_mission_cfg.h.

13.4.1.22 CFE_EVS_MAX_MESSAGE_LENGTH

```
#define CFE_EVS_MAX_MESSAGE_LENGTH CFE_MISSION_EVS_MAX_MESSAGE_LENGTH
```

Definition at line 728 of file sample_mission_cfg.h.

13.4.1.23 CFE_EVS_SEND_HK_MSG

```
#define CFE_EVS_SEND_HK_MSG CFE_MISSION_EVS_SEND_HK_MSG
```

Definition at line 754 of file sample_mission_cfg.h.

13.4.1.24 CFE_MISSION_CMD_APPID_BASE1

```
#define CFE_MISSION_CMD_APPID_BASE1 1
```

Definition at line 389 of file sample_mission_cfg.h.

13.4.1.25 CFE_MISSION_CMD_MID_BASE1

```
#define CFE_MISSION_CMD_MID_BASE1 0x1800
```

Purpose cFE Message ID Base Numbers

Description:

Message Id base numbers for the cFE messages These will now differ in format when using CCSDS version 2 as they will no longer include the Secondary Header Flag and CCSDS version bits.

NOTES: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

For MESSAGE_FORMAT_IS_CCSDS_VER_2 These base MsgIds values are dependent on the values returned by the following SB Macros to form a 16 bit message ID (default macro definitions are in cfe_sb_msg_id_utils.h, default values below are representative of default macro definitions) : CFE_SB_CMD_MESSAGE_TYPE, CFE_SB_RD_APID_FR↩ OM_MSGID CFE_SB_RD_SUBSYS_ID_FROM_MSGID and CFE_SB_RD_TYPE_FROM_MSGID

Limits

Must be less than CFE_PLATFORM_SB_HIGHEST_VALID_MSGID

Definition at line 382 of file sample_mission_cfg.h.

13.4.1.26 CFE_MISSION_CMD_MID_BASE_GLOB

```
#define CFE_MISSION_CMD_MID_BASE_GLOB 0x1860
```

Definition at line 393 of file sample_mission_cfg.h.

13.4.1.27 CFE_MISSION_ES_APP_TLM_MSG

```
#define CFE_MISSION_ES_APP_TLM_MSG 11
```

Definition at line 468 of file sample_mission_cfg.h.

13.4.1.28 CFE_MISSION_ES_CDS_MAX_NAME_LEN

```
#define CFE_MISSION_ES_CDS_MAX_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + CFE_MISSION_MAX_AP↩
I_LEN + 4)
```

Purpose Maximum Length of Full CDS Name in messages

Description:

Indicates the maximum length (in characters) of the entire CDS name of the following form: "ApplicationName.C↩
DSName"

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 683 of file sample_mission_cfg.h.

13.4.1.29 CFE_MISSION_ES_CDS_MAX_NAME_LENGTH

```
#define CFE_MISSION_ES_CDS_MAX_NAME_LENGTH 16
```

Purpose Maximum Length of CDS Name

Description:

Indicates the maximum length (in characters) of the CDS name ('CDSName') portion of a Full CDS Name of the following form: "ApplicationName.CDSName"

Limits

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 307 of file sample_mission_cfg.h.

Referenced by CFE_ES_RegisterCDS().

13.4.1.30 CFE_MISSION_ES_CMD_MSG

```
#define CFE_MISSION_ES_CMD_MSG 6
```

Definition at line 418 of file sample_mission_cfg.h.

13.4.1.31 CFE_MISSION_ES_CRC_16

```
#define CFE_MISSION_ES_CRC_16 2
```

Definition at line 327 of file sample_mission_cfg.h.

Referenced by CFE_ES_CalculateCRC().

13.4.1.32 CFE_MISSION_ES_CRC_32

```
#define CFE_MISSION_ES_CRC_32 3
```

Definition at line 328 of file sample_mission_cfg.h.

Referenced by CFE_ES_CalculateCRC().

13.4.1.33 CFE_MISSION_ES_CRC_8

```
#define CFE_MISSION_ES_CRC_8 1
```

Definition at line 326 of file sample_mission_cfg.h.

Referenced by CFE_ES_CalculateCRC().

13.4.1.34 CFE_MISSION_ES_DEFAULT_CRC

```
#define CFE_MISSION_ES_DEFAULT_CRC CFE_MISSION_ES_CRC_16
```

Purpose Mission Default CRC algorithm

Description:

Indicates the which CRC algorithm should be used as the default for verifying the contents of Critical Data Stores and when calculating Table Image data integrity values.

Limits

Currently only CFE_MISSION_ES_CRC_16 is supported (see CFE_MISSION_ES_CRC_16)

Definition at line 342 of file sample_mission_cfg.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), and CFE_ES_TaskInit().

13.4.1.35 CFE_MISSION_ES_HK_TLM_MSG

```
#define CFE_MISSION_ES_HK_TLM_MSG 0
```

Purpose cFE Portable Message Numbers for Telemetry

Description:

Portable message numbers for the cFE telemetry messages NOTE: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

Limits

Not Applicable

Definition at line 457 of file sample_mission_cfg.h.

13.4.1.36 CFE_MISSION_ES_MAX_APPLICATIONS

```
#define CFE_MISSION_ES_MAX_APPLICATIONS 16
```

Purpose Mission Max Apps in a message

Description:

Indicates the maximum number of apps in a telemetry housekeeping message

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 489 of file sample_mission_cfg.h.

Referenced by CFE_EVS_ReportHousekeepingCmd().

13.4.1.37 CFE_MISSION_ES_MAX_SHELL_CMD

```
#define CFE_MISSION_ES_MAX_SHELL_CMD 64
```

Purpose Define Max Shell Command Size for messages

Description:

Defines the maximum size in characters of the shell command.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 509 of file sample_mission_cfg.h.

13.4.1.38 CFE_MISSION_ES_MAX_SHELL_PKT

```
#define CFE_MISSION_ES_MAX_SHELL_PKT 64
```

Purpose Define Shell Command Telemetry Pkt Segment Size for messages

Description:

Defines the size of the shell command tlm packet segments.The shell command output size is dependant on the shell command itself. If the shell output size is greater than the size of the packet defined here, the fsw will generate a series of tlm packets (of the size defined here) that can be reconstructed by the ground system.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 534 of file sample_mission_cfg.h.

Referenced by CFE_ES_ShellOutputCommand().

13.4.1.39 CFE_MISSION_ES_MEMSTATS_TLM_MSG

```
#define CFE_MISSION_ES_MEMSTATS_TLM_MSG 16
```

Definition at line 473 of file sample_mission_cfg.h.

13.4.1.40 CFE_MISSION_ES_PERF_MAX_IDS

```
#define CFE_MISSION_ES_PERF_MAX_IDS 128
```

Purpose Define Max Number of Performance IDs for messages

Description:

Defines the maximum number of perf ids allowed in command/telemetry messages

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 551 of file sample_mission_cfg.h.

Referenced by CFE_ES_PerfLogAdd().

13.4.1.41 CFE_MISSION_ES_SEND_HK_MSG

```
#define CFE_MISSION_ES_SEND_HK_MSG 8
```

Definition at line 420 of file sample_mission_cfg.h.

13.4.1.42 CFE_MISSION_ES_SHELL_TLM_MSG

```
#define CFE_MISSION_ES_SHELL_TLM_MSG 15
```

Definition at line 472 of file sample_mission_cfg.h.

13.4.1.43 CFE_MISSION_EVS_CMD_MSG

```
#define CFE_MISSION_EVS_CMD_MSG 1
```

Purpose cFE Portable Message Numbers for Commands

Description:

Portable message numbers for the cFE command messages NOTE: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

Limits

Not Applicable

Definition at line 413 of file sample_mission_cfg.h.

13.4.1.44 CFE_MISSION_EVS_HK_TLM_MSG

```
#define CFE_MISSION_EVS_HK_TLM_MSG 1
```

Definition at line 458 of file sample_mission_cfg.h.

13.4.1.45 CFE_MISSION_EVS_LONG_EVENT_MSG_MSG

```
#define CFE_MISSION_EVS_LONG_EVENT_MSG_MSG 8
```

Definition at line 465 of file sample_mission_cfg.h.

13.4.1.46 CFE_MISSION_EVS_MAX_MESSAGE_LENGTH

```
#define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122
```

Purpose Maximum Event Message Length

Description:

Indicates the maximum length (in characters) of the formatted text string portion of an event message

Limits

Not Applicable

Definition at line 321 of file sample_mission_cfg.h.

Referenced by CFE_ES_TaskInit().

13.4.1.47 CFE_MISSION_EVS_SEND_HK_MSG

```
#define CFE_MISSION_EVS_SEND_HK_MSG 9
```

Definition at line 421 of file sample_mission_cfg.h.

13.4.1.48 CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG

```
#define CFE_MISSION_EVS_SHORT_EVENT_MSG_MSG 9
```

Definition at line 466 of file sample_mission_cfg.h.

13.4.1.49 CFE_MISSION_MAX_API_LEN

```
#define CFE_MISSION_MAX_API_LEN 20
```

Purpose cFE Maximum length for API names within data exchange structures

Description:

The value of this constant dictates the size of filenames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS_MAX_API_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS_MAX_API_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS_MAX_API_LEN value.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 663 of file sample_mission_cfg.h.

13.4.1.50 CFE_MISSION_MAX_FILE_LEN

```
#define CFE_MISSION_MAX_FILE_LEN 20
```

Purpose cFE Maximum length for filenames within data exchange structures

Description:

The value of this constant dictates the size of filenames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS_MAX_FILE_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS_MAX_FILE_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS_MAX_FILE_LEN value.

Limits

All CPUs within the same SB domain (mission) and ground tools must share the same definition. Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 639 of file sample_mission_cfg.h.

13.4.1.51 CFE_MISSION_MAX_PATH_LEN

```
#define CFE_MISSION_MAX_PATH_LEN 64
```

Purpose cFE Maximum length for pathnames within data exchange structures

Description:

The value of this constant dictates the size of pathnames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS_MAX_PATH_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS_MAX_PATH_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS_MAX_PATH_LEN value.

Limits

All CPUs within the same SB domain (mission) and ground tools must share the same definition. Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 614 of file sample_mission_cfg.h.

13.4.1.52 CFE_MISSION_SB_ALLSUBS_TLM_MSG

```
#define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13
```

Definition at line 470 of file sample_mission_cfg.h.

13.4.1.53 CFE_MISSION_SB_CMD_MSG

```
#define CFE_MISSION_SB_CMD_MSG 3
```

Definition at line 415 of file sample_mission_cfg.h.

13.4.1.54 CFE_MISSION_SB_HK_TLM_MSG

```
#define CFE_MISSION_SB_HK_TLM_MSG 3
```

Definition at line 460 of file sample_mission_cfg.h.

13.4.1.55 CFE_MISSION_SB_MAX_PIPES

```
#define CFE_MISSION_SB_MAX_PIPES 64
```

Purpose Maximum Number of pipes that SB command/telemetry messages may hold

Description:

Dictates the maximum number of unique Pipes the SB message defintions will hold.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 588 of file sample_mission_cfg.h.

13.4.1.56 CFE_MISSION_SB_MAX_SB_MSG_SIZE

```
#define CFE_MISSION_SB_MAX_SB_MSG_SIZE 32768
```

Purpose Maximum SB Message Size

Description:

The following definition dictates the maximum message size allowed on the software bus. SB checks the pkt length field in the header of all messages sent. If the pkt length field indicates the message is larger than this define, SB sends an event and rejects the send.

Limits

This parameter has a lower limit of 6 (CCSDS primary header size). There are no restrictions on the upper limit however, the maximum message size is system dependent and should be verified. Total message size values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 108 of file sample_mission_cfg.h.

Referenced by CFE_SB_SendMsgFull().

13.4.1.57 CFE_MISSION_SB_ONESUB_TLM_MSG

```
#define CFE_MISSION_SB_ONESUB_TLM_MSG 14
```

Definition at line 471 of file sample_mission_cfg.h.

13.4.1.58 CFE_MISSION_SB_PACKET_TIME_FORMAT

```
#define CFE_MISSION_SB_PACKET_TIME_FORMAT CFE_MISSION_SB_TIME_32_16_SUBS
```

Purpose Packet Timestamp Format Selection

Description:

Defines the size, format and contents of the telemetry packet timestamp.

Limits

Must be defined as one of the supported formats listed above

Definition at line 89 of file sample_mission_cfg.h.

13.4.1.59 CFE_MISSION_SB_SEND_HK_MSG

```
#define CFE_MISSION_SB_SEND_HK_MSG 11
```

Definition at line 423 of file sample_mission_cfg.h.

13.4.1.60 CFE_MISSION_SB_STATS_TLM_MSG

```
#define CFE_MISSION_SB_STATS_TLM_MSG 10
```

Definition at line 467 of file sample_mission_cfg.h.

13.4.1.61 CFE_MISSION_SB_TIME_32_16_SUBS

```
#define CFE_MISSION_SB_TIME_32_16_SUBS 1
```

Definition at line 75 of file sample_mission_cfg.h.

13.4.1.62 CFE_MISSION_SB_TIME_32_32_M_20

```
#define CFE_MISSION_SB_TIME_32_32_M_20 3
```

Definition at line 77 of file sample_mission_cfg.h.

13.4.1.63 CFE_MISSION_SB_TIME_32_32_SUBS

```
#define CFE_MISSION_SB_TIME_32_32_SUBS 2
```

Definition at line 76 of file sample_mission_cfg.h.

13.4.1.64 CFE_MISSION_SPACECRAFT_ID

```
#define CFE_MISSION_SPACECRAFT_ID 0x42
```

Purpose Spacecraft ID

Description:

This defines the value that is returned by the call to CFE_PSP_GetSpacecraftId.

Limits

The cFE does not place a limit on this configuration paramter. CCSDS allocates 8 bits for this field in the standard VCDU.

Definition at line 52 of file sample_mission_cfg.h.

13.4.1.65 CFE_MISSION_TBL_CMD_MSG

```
#define CFE_MISSION_TBL_CMD_MSG 4
```

Definition at line 416 of file sample_mission_cfg.h.

13.4.1.66 CFE_MISSION_TBL_HK_TLM_MSG

```
#define CFE_MISSION_TBL_HK_TLM_MSG 4
```

Definition at line 461 of file sample_mission_cfg.h.

13.4.1.67 CFE_MISSION_TBL_MAX_FULL_NAME_LEN

```
#define CFE_MISSION_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_MISSION_MAX_API↩
_LEN + 4)
```

Purpose Maximum Length of Full Table Name in messages

Description:

Indicates the maximum length (in characters) of the entire table name within software bus messages, in "App↩ Name.TableName" notation.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 571 of file sample_mission_cfg.h.

13.4.1.68 CFE_MISSION_TBL_MAX_NAME_LENGTH

```
#define CFE_MISSION_TBL_MAX_NAME_LENGTH 16
```

Purpose Maximum Table Name Length

Description:

Indicates the maximum length (in characers) of the table name ('TblName') portion of a Full Table Name of the following form: "ApplicationName.TblName"

Limits

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 357 of file sample_mission_cfg.h.

13.4.1.69 CFE_MISSION_TBL_REG_TLM_MSG

```
#define CFE_MISSION_TBL_REG_TLM_MSG 12
```

Definition at line 469 of file sample_mission_cfg.h.

13.4.1.70 CFE_MISSION_TBL_SEND_HK_MSG

```
#define CFE_MISSION_TBL_SEND_HK_MSG 12
```

Definition at line 424 of file sample_mission_cfg.h.

13.4.1.71 CFE_MISSION_TIME_1HZ_CMD_MSG

```
#define CFE_MISSION_TIME_1HZ_CMD_MSG 17
```

Definition at line 428 of file sample_mission_cfg.h.

13.4.1.72 CFE_MISSION_TIME_AT_TONE_WAS

```
#define CFE_MISSION_TIME_AT_TONE_WAS true
```

Purpose Default Time and Tone Order

Description:

Time Services may be configured to expect the time at the tone data packet to either precede or follow the tone signal. If the time at the tone data packet follows the tone signal, then the data within the packet describes what the time "was" at the tone. If the time at the tone data packet precedes the tone signal, then the data within the packet describes what the time "will be" at the tone. One, and only one, of the following symbols must be set to true:

- CFE_MISSION_TIME_AT_TONE_WAS
- CFE_MISSION_TIME_AT_TONE_WILL_BE Note: If Time Services is defined as using a simulated tone signal (see CFE_MISSION_TIME_CFG_FAKE_TONE above), then the tone data packet must follow the tone signal.

Limits

Either CFE_MISSION_TIME_AT_TONE_WAS or CFE_MISSION_TIME_AT_TONE_WILL_BE must be set to true. They may not both be true and they may not both be false.

Definition at line 168 of file sample_mission_cfg.h.

13.4.1.73 CFE_MISSION_TIME_AT_TONE_WILL_BE

```
#define CFE_MISSION_TIME_AT_TONE_WILL_BE false
```

Definition at line 169 of file sample_mission_cfg.h.

13.4.1.74 CFE_MISSION_TIME_CFG_DEFAULT_TAI

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI true
```

Purpose Default Time Format

Description:

The following definitions select either UTC or TAI as the default (mission specific) time format. Although it is possible for an application to request time in a specific format, most callers should use CFE_TIME_GetTime(), which returns time in the default format. This avoids having to modify each individual caller when the default choice is changed.

Limits

if CFE_MISSION_TIME_CFG_DEFAULT_TAI is defined as true then CFE_MISSION_TIME_CFG_DEFAULT_UTC must be defined as false. if CFE_MISSION_TIME_CFG_DEFAULT_TAI is defined as false then CFE_MISSION↩ _TIME_CFG_DEFAULT_UTC must be defined as true.

Definition at line 129 of file sample_mission_cfg.h.

13.4.1.75 CFE_MISSION_TIME_CFG_DEFAULT_UTC

```
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC false
```

Definition at line 130 of file sample_mission_cfg.h.

13.4.1.76 CFE_MISSION_TIME_CFG_FAKE_TONE

```
#define CFE_MISSION_TIME_CFG_FAKE_TONE true
```

Purpose Default Time Format

Description:

The following definition enables the use of a simulated time at the tone signal using a software bus message.

Limits

Not Applicable

Definition at line 144 of file sample_mission_cfg.h.

13.4.1.77 CFE_MISSION_TIME_CMD_MSG

```
#define CFE_MISSION_TIME_CMD_MSG 5
```

Definition at line 417 of file sample_mission_cfg.h.

13.4.1.78 CFE_MISSION_TIME_DATA_CMD_MSG

```
#define CFE_MISSION_TIME_DATA_CMD_MSG 0
```

Purpose cFE Portable Message Numbers for Global Messages

Description:

Portable message numbers for the cFE global messages NOTE: cFE MsgIds are the sum of the base numbers and the portable msg numbers.

Limits

Not Applicable

Definition at line 442 of file sample_mission_cfg.h.

13.4.1.79 CFE_MISSION_TIME_DEF_DELAY_SECS

```
#define CFE_MISSION_TIME_DEF_DELAY_SECS 0
```

Definition at line 229 of file sample_mission_cfg.h.

13.4.1.80 CFE_MISSION_TIME_DEF_DELAY_SUBS

```
#define CFE_MISSION_TIME_DEF_DELAY_SUBS 1000
```

Definition at line 230 of file sample_mission_cfg.h.

13.4.1.81 CFE_MISSION_TIME_DEF_LEAPS

```
#define CFE_MISSION_TIME_DEF_LEAPS 32
```

Definition at line 227 of file sample_mission_cfg.h.

13.4.1.82 CFE_MISSION_TIME_DEF_MET_SECS

```
#define CFE_MISSION_TIME_DEF_MET_SECS 1000
```

Purpose Default Time Values

Description:

Default time values are provided to avoid problems due to time calculations performed after startup but before commands can be processed. For example, if the default time format is UTC then it is important that the sum of MET and STCF always exceed the value of Leap Seconds to prevent the UTC time calculation `(time = MET + STCF - Leap Seconds)` from resulting in a negative (very large) number.

Some past missions have also created known (albeit wrong) default timestamps. For example, assume the epoch is defined as Jan 1, 1970 and further assume the default time values are set to create a timestamp of Jan 1, 2000. Even though the year 2000 timestamps are wrong, it may be of value to keep the time within some sort of bounds acceptable to the software.

Note: Sub-second units are in micro-seconds (0 to 999,999) and all values must be defined

Limits

Not Applicable

Definition at line 221 of file sample_mission_cfg.h.

13.4.1.83 CFE_MISSION_TIME_DEF_MET_SUBS

```
#define CFE_MISSION_TIME_DEF_MET_SUBS 0
```

Definition at line 222 of file sample_mission_cfg.h.

13.4.1.84 CFE_MISSION_TIME_DEF_STCF_SECS

```
#define CFE_MISSION_TIME_DEF_STCF_SECS 1000000
```

Definition at line 224 of file sample_mission_cfg.h.

13.4.1.85 CFE_MISSION_TIME_DEF_STCF_SUBS

```
#define CFE_MISSION_TIME_DEF_STCF_SUBS 0
```

Definition at line 225 of file sample_mission_cfg.h.

13.4.1.86 CFE_MISSION_TIME_DIAG_TLM_MSG

```
#define CFE_MISSION_TIME_DIAG_TLM_MSG 6
```

Definition at line 463 of file sample_mission_cfg.h.

13.4.1.87 CFE_MISSION_TIME_EPOCH_DAY

```
#define CFE_MISSION_TIME_EPOCH_DAY 1
```

Definition at line 248 of file sample_mission_cfg.h.

13.4.1.88 CFE_MISSION_TIME_EPOCH_HOUR

```
#define CFE_MISSION_TIME_EPOCH_HOUR 0
```

Definition at line 249 of file sample_mission_cfg.h.

13.4.1.89 CFE_MISSION_TIME_EPOCH_MINUTE

```
#define CFE_MISSION_TIME_EPOCH_MINUTE 0
```

Definition at line 250 of file sample_mission_cfg.h.

13.4.1.90 CFE_MISSION_TIME_EPOCH_SECOND

```
#define CFE_MISSION_TIME_EPOCH_SECOND 0
```

Definition at line 251 of file sample_mission_cfg.h.

13.4.1.91 CFE_MISSION_TIME_EPOCH_YEAR

```
#define CFE_MISSION_TIME_EPOCH_YEAR 1980
```

Purpose Default EPOCH Values

Description:

Default ground time epoch values Note: these values are used only by the CFE_TIME_Print() API function

Limits

Year - must be within 136 years Day - Jan 1 = 1, Feb 1 = 32, etc. Hour - 0 to 23 Minute - 0 to 59 Second - 0 to 59

Definition at line 247 of file sample_mission_cfg.h.

13.4.1.92 CFE_MISSION_TIME_FS_FACTOR

```
#define CFE_MISSION_TIME_FS_FACTOR 789004800
```

Purpose Time File System Factor

Description:

Define the s/c vs file system time conversion constant...

Note: this value is intended for use only by CFE TIME API functions to convert time values based on the ground system epoch (s/c time) to and from time values based on the file system epoch (fs time).

FS time = S/C time + factor S/C time = FS time - factor

Worksheet:

S/C epoch = Jan 1, 2005 (LRO ground system epoch) FS epoch = Jan 1, 1980 (vxWorks DOS file system epoch)

Delta = 25 years, 0 days, 0 hours, 0 minutes, 0 seconds

Leap years = 1980, 1984, 1988, 1992, 1996, 2000, 2004 (divisible by 4 – except if by 100 – unless also by 400)

1 year = 31,536,000 seconds 1 day = 86,400 seconds 1 hour = 3,600 seconds 1 minute = 60 seconds

25 years = 788,400,000 seconds 7 extra leap days = 604,800 seconds

total delta = 789,004,800 seconds

Limits

Not Applicable

Definition at line 290 of file sample_mission_cfg.h.

13.4.1.93 CFE_MISSION_TIME_HK_TLM_MSG

```
#define CFE_MISSION_TIME_HK_TLM_MSG 5
```

Definition at line 462 of file sample_mission_cfg.h.

13.4.1.94 CFE_MISSION_TIME_MAX_ELAPSED

```
#define CFE_MISSION_TIME_MAX_ELAPSED 200000
```

Definition at line 195 of file sample_mission_cfg.h.

13.4.1.95 CFE_MISSION_TIME_MIN_ELAPSED

```
#define CFE_MISSION_TIME_MIN_ELAPSED 0
```

Purpose Min and Max Time Elapsed

Description:

Based on the definition of Time and Tone Order (CFE_MISSION_TIME_AT_TONE_WAS/WILL_BE) either the "time at the tone" signal or data packet will follow the other. This definition sets the valid window of time for the second of the pair to lag behind the first. Time Services will invalidate both the tone and packet if the second does not arrive within this window following the first.

For example, if the data packet follows the tone, it might be valid for the data packet to arrive between zero and 100,000 micro-seconds after the tone. But, if the tone follows the the packet, it might be valid only if the packet arrived between 200,000 and 700,000 micro-seconds before the tone.

Note: units are in micro-seconds

Limits

0 to 999,999 decimal

Definition at line 194 of file sample_mission_cfg.h.

13.4.1.96 CFE_MISSION_TIME_SEND_CMD_MSG

```
#define CFE_MISSION_TIME_SEND_CMD_MSG 2
```

Definition at line 443 of file sample_mission_cfg.h.

13.4.1.97 CFE_MISSION_TIME_SEND_HK_MSG

```
#define CFE_MISSION_TIME_SEND_HK_MSG 13
```

Definition at line 425 of file sample_mission_cfg.h.

13.4.1.98 CFE_MISSION_TIME_TONE_CMD_MSG

```
#define CFE_MISSION_TIME_TONE_CMD_MSG 16
```

Definition at line 427 of file sample_mission_cfg.h.

13.4.1.99 CFE_MISSION_TLM_APPID_BASE1

```
#define CFE_MISSION_TLM_APPID_BASE1 0
```

Definition at line 390 of file sample_mission_cfg.h.

13.4.1.100 CFE_MISSION_TLM_MID_BASE1

```
#define CFE_MISSION_TLM_MID_BASE1 0x0800
```

Definition at line 383 of file sample_mission_cfg.h.

13.4.1.101 CFE_MISSION_TLM_MID_BASE_GLOB

```
#define CFE_MISSION_TLM_MID_BASE_GLOB 0x0860
```

Definition at line 394 of file sample_mission_cfg.h.

13.4.1.102 CFE_SB_ALLSUBS_TLM_MSG

```
#define CFE_SB_ALLSUBS_TLM_MSG CFE_MISSION_SB_ALLSUBS_TLM_MSG
```

Definition at line 772 of file sample_mission_cfg.h.

13.4.1.103 CFE_SB_CMD_MSG

```
#define CFE_SB_CMD_MSG CFE_MISSION_SB_CMD_MSG
```

Definition at line 749 of file sample_mission_cfg.h.

13.4.1.104 CFE_SB_HK_TLM_MSG

```
#define CFE_SB_HK_TLM_MSG CFE_MISSION_SB_HK_TLM_MSG
```

Definition at line 764 of file sample_mission_cfg.h.

13.4.1.105 CFE_SB_MAX_SB_MSG_SIZE

```
#define CFE_SB_MAX_SB_MSG_SIZE CFE_MISSION_SB_MAX_SB_MSG_SIZE
```

Definition at line 706 of file sample_mission_cfg.h.

13.4.1.106 CFE_SB_ONESUB_TLM_MSG

```
#define CFE_SB_ONESUB_TLM_MSG CFE_MISSION_SB_ONESUB_TLM_MSG
```

Definition at line 773 of file sample_mission_cfg.h.

13.4.1.107 CFE_SB_PACKET_TIME_FORMAT

```
#define CFE_SB_PACKET_TIME_FORMAT CFE_MISSION_SB_PACKET_TIME_FORMAT
```

Definition at line 705 of file sample_mission_cfg.h.

13.4.1.108 CFE_SB_SEND_HK_MSG

```
#define CFE_SB_SEND_HK_MSG CFE_MISSION_SB_SEND_HK_MSG
```

Definition at line 755 of file sample_mission_cfg.h.

13.4.1.109 CFE_SB_STATS_TLM_MSG

```
#define CFE_SB_STATS_TLM_MSG CFE_MISSION_SB_STATS_TLM_MSG
```

Definition at line 769 of file sample_mission_cfg.h.

13.4.1.110 CFE_SB_TIME_32_16_SUBS

```
#define CFE_SB_TIME_32_16_SUBS CFE_MISSION_SB_TIME_32_16_SUBS
```

Definition at line 702 of file sample_mission_cfg.h.

13.4.1.111 CFE_SB_TIME_32_32_M_20

```
#define CFE_SB_TIME_32_32_M_20 CFE_MISSION_SB_TIME_32_32_M_20
```

Definition at line 704 of file sample_mission_cfg.h.

13.4.1.112 CFE_SB_TIME_32_32_SUBS

```
#define CFE_SB_TIME_32_32_SUBS CFE_MISSION_SB_TIME_32_32_SUBS
```

Definition at line 703 of file sample_mission_cfg.h.

13.4.1.113 CFE_SPACECRAFT_ID

```
#define CFE_SPACECRAFT_ID CFE_MISSION_SPACECRAFT_ID
```

Definition at line 701 of file sample_mission_cfg.h.

Referenced by CFE_SB_SetMsgId().

13.4.1.114 CFE_TBL_CMD_MSG

```
#define CFE_TBL_CMD_MSG CFE_MISSION_TBL_CMD_MSG
```

Definition at line 750 of file sample_mission_cfg.h.

13.4.1.115 CFE_TBL_HK_TLM_MSG

```
#define CFE_TBL_HK_TLM_MSG CFE_MISSION_TBL_HK_TLM_MSG
```

Definition at line 765 of file sample_mission_cfg.h.

13.4.1.116 CFE_TBL_MAX_NAME_LENGTH

```
#define CFE_TBL_MAX_NAME_LENGTH CFE_MISSION_TBL_MAX_NAME_LENGTH
```

Definition at line 733 of file sample_mission_cfg.h.

13.4.1.117 CFE_TBL_REG_TLM_MSG

```
#define CFE_TBL_REG_TLM_MSG CFE_MISSION_TBL_REG_TLM_MSG
```

Definition at line 771 of file sample_mission_cfg.h.

13.4.1.118 CFE_TBL_SEND_HK_MSG

```
#define CFE_TBL_SEND_HK_MSG CFE_MISSION_TBL_SEND_HK_MSG
```

Definition at line 756 of file sample_mission_cfg.h.

13.4.1.119 CFE_TIME_1HZ_CMD_MSG

```
#define CFE_TIME_1HZ_CMD_MSG CFE_MISSION_TIME_1HZ_CMD_MSG
```

Definition at line 759 of file sample_mission_cfg.h.

13.4.1.120 CFE_TIME_AT_TONE_WAS

```
#define CFE_TIME_AT_TONE_WAS CFE_MISSION_TIME_AT_TONE_WAS
```

Definition at line 710 of file sample_mission_cfg.h.

13.4.1.121 CFE_TIME_AT_TONE_WILL_BE

```
#define CFE_TIME_AT_TONE_WILL_BE CFE_MISSION_TIME_AT_TONE_WILL_BE
```

Definition at line 711 of file sample_mission_cfg.h.

13.4.1.122 CFE_TIME_CFG_DEFAULT_TAI

```
#define CFE_TIME_CFG_DEFAULT_TAI CFE_MISSION_TIME_CFG_DEFAULT_TAI
```

Definition at line 707 of file sample_mission_cfg.h.

13.4.1.123 CFE_TIME_CFG_DEFAULT_UTC

```
#define CFE_TIME_CFG_DEFAULT_UTC CFE_MISSION_TIME_CFG_DEFAULT_UTC
```

Definition at line 708 of file sample_mission_cfg.h.

13.4.1.124 CFE_TIME_CFG_FAKE_TONE

```
#define CFE_TIME_CFG_FAKE_TONE CFE_MISSION_TIME_CFG_FAKE_TONE
```

Definition at line 709 of file sample_mission_cfg.h.

13.4.1.125 CFE_TIME_CMD_MSG

```
#define CFE_TIME_CMD_MSG CFE_MISSION_TIME_CMD_MSG
```

Definition at line 751 of file sample_mission_cfg.h.

13.4.1.126 CFE_TIME_DATA_CMD_MSG

```
#define CFE_TIME_DATA_CMD_MSG CFE_MISSION_TIME_DATA_CMD_MSG
```

Definition at line 760 of file sample_mission_cfg.h.

13.4.1.127 CFE_TIME_DEF_DELAY_SECS

```
#define CFE_TIME_DEF_DELAY_SECS CFE_MISSION_TIME_DEF_DELAY_SECS
```

Definition at line 719 of file sample_mission_cfg.h.

13.4.1.128 CFE_TIME_DEF_DELAY_SUBS

```
#define CFE_TIME_DEF_DELAY_SUBS CFE_MISSION_TIME_DEF_DELAY_SUBS
```

Definition at line 720 of file sample_mission_cfg.h.

13.4.1.129 CFE_TIME_DEF_LEAPS

```
#define CFE_TIME_DEF_LEAPS CFE_MISSION_TIME_DEF_LEAPS
```

Definition at line 718 of file sample_mission_cfg.h.

13.4.1.130 CFE_TIME_DEF_MET_SECS

```
#define CFE_TIME_DEF_MET_SECS CFE_MISSION_TIME_DEF_MET_SECS
```

Definition at line 714 of file sample_mission_cfg.h.

13.4.1.131 CFE_TIME_DEF_MET_SUBS

```
#define CFE_TIME_DEF_MET_SUBS CFE_MISSION_TIME_DEF_MET_SUBS
```

Definition at line 715 of file sample_mission_cfg.h.

13.4.1.132 CFE_TIME_DEF_STCF_SECS

```
#define CFE_TIME_DEF_STCF_SECS CFE_MISSION_TIME_DEF_STCF_SECS
```

Definition at line 716 of file sample_mission_cfg.h.

13.4.1.133 CFE_TIME_DEF_STCF_SUBS

```
#define CFE_TIME_DEF_STCF_SUBS CFE_MISSION_TIME_DEF_STCF_SUBS
```

Definition at line 717 of file sample_mission_cfg.h.

13.4.1.134 CFE_TIME_DIAG_TLM_MSG

```
#define CFE_TIME_DIAG_TLM_MSG CFE_MISSION_TIME_DIAG_TLM_MSG
```

Definition at line 767 of file sample_mission_cfg.h.

13.4.1.135 CFE_TIME_EPOCH_DAY

```
#define CFE_TIME_EPOCH_DAY CFE_MISSION_TIME_EPOCH_DAY
```

Definition at line 722 of file sample_mission_cfg.h.

13.4.1.136 CFE_TIME_EPOCH_HOUR

```
#define CFE_TIME_EPOCH_HOUR CFE_MISSION_TIME_EPOCH_HOUR
```

Definition at line 723 of file sample_mission_cfg.h.

13.4.1.137 CFE_TIME_EPOCH_MINUTE

```
#define CFE_TIME_EPOCH_MINUTE CFE_MISSION_TIME_EPOCH_MINUTE
```

Definition at line 724 of file sample_mission_cfg.h.

13.4.1.138 CFE_TIME_EPOCH_SECOND

```
#define CFE_TIME_EPOCH_SECOND CFE_MISSION_TIME_EPOCH_SECOND
```

Definition at line 725 of file sample_mission_cfg.h.

13.4.1.139 CFE_TIME_EPOCH_YEAR

```
#define CFE_TIME_EPOCH_YEAR CFE_MISSION_TIME_EPOCH_YEAR
```

Definition at line 721 of file sample_mission_cfg.h.

13.4.1.140 CFE_TIME_FS_FACTOR

```
#define CFE_TIME_FS_FACTOR CFE_MISSION_TIME_FS_FACTOR
```

Definition at line 726 of file sample_mission_cfg.h.

13.4.1.141 CFE_TIME_HK_TLM_MSG

```
#define CFE_TIME_HK_TLM_MSG CFE_MISSION_TIME_HK_TLM_MSG
```

Definition at line 766 of file sample_mission_cfg.h.

13.4.1.142 CFE_TIME_MAX_ELAPSED

```
#define CFE_TIME_MAX_ELAPSED CFE_MISSION_TIME_MAX_ELAPSED
```

Definition at line 713 of file sample_mission_cfg.h.

13.4.1.143 CFE_TIME_MIN_ELAPSED

```
#define CFE_TIME_MIN_ELAPSED CFE_MISSION_TIME_MIN_ELAPSED
```

Definition at line 712 of file sample_mission_cfg.h.

13.4.1.144 CFE_TIME_SEND_CMD_MSG

```
#define CFE_TIME_SEND_CMD_MSG CFE_MISSION_TIME_SEND_CMD_MSG
```

Definition at line 761 of file sample_mission_cfg.h.

13.4.1.145 CFE_TIME_SEND_HK_MSG

```
#define CFE_TIME_SEND_HK_MSG CFE_MISSION_TIME_SEND_HK_MSG
```

Definition at line 757 of file sample_mission_cfg.h.

13.4.1.146 CFE_TIME_TONE_CMD_MSG

```
#define CFE_TIME_TONE_CMD_MSG CFE_MISSION_TIME_TONE_CMD_MSG
```

Definition at line 758 of file sample_mission_cfg.h.

13.4.1.147 CFE_TLM_APPID_BASE_CPU1

```
#define CFE_TLM_APPID_BASE_CPU1 CFE_MISSION_TLM_APPID_BASE_CPU1
```

Definition at line 737 of file sample_mission_cfg.h.

13.4.1.148 CFE_TLM_APPID_BASE_CPU2

```
#define CFE_TLM_APPID_BASE_CPU2 CFE_MISSION_TLM_APPID_BASE_CPU2
```

Definition at line 741 of file sample_mission_cfg.h.

13.4.1.149 CFE_TLM_APPID_BASE_CPU3

```
#define CFE_TLM_APPID_BASE_CPU3 CFE_MISSION_TLM_APPID_BASE_CPU3
```

Definition at line 745 of file sample_mission_cfg.h.

13.4.1.150 CFE_TLM_MID_BASE_CPU1

```
#define CFE_TLM_MID_BASE_CPU1 CFE_MISSION_TLM_MID_BASE_CPU1
```

Definition at line 735 of file sample_mission_cfg.h.

13.4.1.151 CFE_TLM_MID_BASE_CPU2

```
#define CFE_TLM_MID_BASE_CPU2 CFE_MISSION_TLM_MID_BASE_CPU2
```

Definition at line 739 of file sample_mission_cfg.h.

13.4.1.152 CFE_TLM_MID_BASE_CPU3

```
#define CFE_TLM_MID_BASE_CPU3 CFE_MISSION_TLM_MID_BASE_CPU3
```

Definition at line 743 of file sample_mission_cfg.h.

13.4.1.153 CFE_TLM_MID_BASE_GLOB

```
#define CFE_TLM_MID_BASE_GLOB CFE_MISSION_TLM_MID_BASE_GLOB
```

Definition at line 747 of file sample_mission_cfg.h.

13.4.1.154 MESSAGE_FORMAT_IS_CCSDS

```
#define MESSAGE_FORMAT_IS_CCSDS
```

Purpose cFE SB message format

Description:

Dictates the message format used by the cFE.

Limits

All versions of the cFE currently support only CCSDS as the message format Defining only MESSAGE_FORMA↩ T_IS_CCSDS implements the 11 bit APID format in the primary header Also defining MESSAGE_FORMAT_IS_↩ CCSDS_VER_2 implements the APID extended header format MESSAGE_FORMAT_IS_CCSDS must be defined for all cFE deployments. MESSAGE_FORMAT_IS_CCSDS_VER_2 is optional

Definition at line 67 of file sample_mission_cfg.h.

13.5 sample_perfids.h File Reference

Macros

- #define CFE_MISSION_ES_PERF_EXIT_BIT 31
  bit (31) is reserved by the perf utilities

cFE Performance Monitor IDs (Reserved IDs 0-31)

- #define CFE_MISSION_ES_MAIN_PERF_ID 1
  Performance ID for Executive Services Task.
- #define CFE_MISSION_EVS_MAIN_PERF_ID 2
  Performance ID for Events Services Task.
- #define CFE_MISSION_TBL_MAIN_PERF_ID 3
  Performance ID for Table Services Task.
- #define CFE_MISSION_SB_MAIN_PERF_ID 4
  Performance ID for Software Bus Services Task.
- #define CFE_MISSION_SB_MSG_LIM_PERF_ID 5
  Performance ID for Software Bus Msg Limit Errors.
- #define CFE_MISSION_SB_PIPE_OFLOW_PERF_ID 27
  Performance ID for Software Bus Pipe Overflow Errors.
- #define CFE_MISSION_TIME_MAIN_PERF_ID 6
  Performance ID for Time Services Task.
- #define CFE_MISSION_TIME_TONE1HZISR_PERF_ID 7
  Performance ID for 1 Hz Tone ISR.
- #define CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID 8
  Performance ID for 1 Hz Local ISR.
- #define CFE_MISSION_TIME_SENDMET_PERF_ID 9
  Performance ID for Time ToneSendMET.
- #define CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID 10
  Performance ID for 1 Hz Local Task.
- #define CFE_MISSION_TIME_TONE1HZTASK_PERF_ID 11
  Performance ID for 1 Hz Tone Task.

13.5.1 Macro Definition Documentation

13.5.1.1 CFE_MISSION_ES_MAIN_PERF_ID

```
#define CFE_MISSION_ES_MAIN_PERF_ID 1
```

Definition at line 45 of file sample_perfids.h.

Referenced by CFE_ES_TaskMain().

13.5.1.2 CFE_MISSION_ES_PERF_EXIT_BIT

```
#define CFE_MISSION_ES_PERF_EXIT_BIT 31
```

Definition at line 41 of file sample_perfids.h.

Referenced by CFE_ES_PerfLogAdd().

13.5.1.3 CFE_MISSION_EVS_MAIN_PERF_ID

```
#define CFE_MISSION_EVS_MAIN_PERF_ID 2
```

Definition at line 46 of file sample_perfids.h.

Referenced by CFE_EVS_TaskMain().

13.5.1.4 CFE_MISSION_SB_MAIN_PERF_ID

```
#define CFE_MISSION_SB_MAIN_PERF_ID 4
```

Definition at line 48 of file sample_perfids.h.

Referenced by CFE_SB_TaskMain().

13.5.1.5 CFE_MISSION_SB_MSG_LIM_PERF_ID

```
#define CFE_MISSION_SB_MSG_LIM_PERF_ID 5
```

Definition at line 49 of file sample_perfids.h.

Referenced by CFE_SB_SendMsgFull().

13.5.1.6 CFE_MISSION_SB_PIPE_OFLOW_PERF_ID

```
#define CFE_MISSION_SB_PIPE_OFLOW_PERF_ID 27
```

Definition at line 50 of file sample_perfids.h.

Referenced by CFE_SB_SendMsgFull().

13.5.1.7 CFE_MISSION_TBL_MAIN_PERF_ID

```
#define CFE_MISSION_TBL_MAIN_PERF_ID 3
```

Definition at line 47 of file sample_perfids.h.

13.5.1.8 CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID

```
#define CFE_MISSION_TIME_LOCAL1HZISR_PERF_ID 8
```

Definition at line 55 of file sample_perfids.h.

13.5.1.9 CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID

```
#define CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID 10
```

Definition at line 58 of file sample_perfids.h.

13.5.1.10 CFE_MISSION_TIME_MAIN_PERF_ID

```
#define CFE_MISSION_TIME_MAIN_PERF_ID 6
```

Definition at line 53 of file sample_perfids.h.

13.5.1.11 CFE_MISSION_TIME_SENDMET_PERF_ID

```
#define CFE_MISSION_TIME_SENDMET_PERF_ID 9
```

Definition at line 57 of file sample_perfids.h.

13.5.1.12 CFE_MISSION_TIME_TONE1HZISR_PERF_ID

```
#define CFE_MISSION_TIME_TONE1HZISR_PERF_ID 7
```

Definition at line 54 of file sample_perfids.h.

13.5.1.13 CFE_MISSION_TIME_TONE1HZTASK_PERF_ID

```
#define CFE_MISSION_TIME_TONE1HZTASK_PERF_ID 11
```

Definition at line 59 of file sample_perfids.h.

13.6 cfe/docs/src/cfe_es.dox File Reference

13.7 cfe/docs/src/cfe_evs.dox File Reference

13.8 cfe/docs/src/cfe_sb.dox File Reference

13.9 cfe/docs/src/cfe_tbl.dox File Reference

13.10 cfe/docs/src/cfe_time.dox File Reference

13.11 cfe/docs/src/cfe_usersguide.dox File Reference

13.12 cfe/docs/src/cfe_xref.dox File Reference

13.13 cfe/docs/src/main.dox File Reference

13.14 cfe/fsw/cfe-core/src/es/cfe_es.mak File Reference

13.15 cfe/fsw/cfe-core/src/es/cfe_es_api.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es.h"
#include "cfe_es_apps.h"
#include "cfe_es_global.h"
#include "cfe_es_events.h"
#include "cfe_es_cds.h"
#include "cfe_es_cds_mempool.h"
#include "cfe_psp.h"
#include "cfe_es_log.h"
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
```

Functions

- int32 CFE_ES_GetResetType (uint32 *ResetSubtypePtr)

  Return the most recent Reset Type.
- int32 CFE_ES_ResetCFE (uint32 ResetType)

  Reset the cFE Core and all cFE Applications.
- void CFE_ES_SetAppState (uint32 AppID, uint32 TargetState)
- int32 CFE_ES_RestartApp (uint32 AppID)

  Restart a single cFE Application.
- int32 CFE_ES_ReloadApp (uint32 AppID, const char *AppFileName)

  Reload a single cFE Application.
- int32 CFE_ES_DeleteApp (uint32 AppID)

  Delete a cFE Application.
- void CFE_ES_ExitApp (uint32 ExitStatus)

  Exit a cFE Application.
- bool CFE_ES_RunLoop (uint32 *RunStatus)

  Check for Exit, Restart, or Reload commands.
- int32 CFE_ES_WaitForSystemState (uint32 MinSystemState, uint32 TimeOutMilliseconds)

  Allow an Application to Wait for a minimum global system state.
- void CFE_ES_WaitForStartupSync (uint32 TimeOutMilliseconds)

  Allow an Application to Wait for the "OPERATIONAL" global system state.
- int32 CFE_ES_RegisterApp (void)

  Registers a cFE Application with the Executive Services.
- int32 CFE_ES_GetAppIDByName (uint32 *AppIdPtr, const char *AppName)

  Get an Application ID associated with a specified Application name.
- int32 CFE_ES_GetAppID (uint32 *AppIdPtr)

  Get an Application ID for the calling Application.
- int32 CFE_ES_GetAppName (char *AppName, uint32 AppId, uint32 BufferLength)

  Get an Application name for a specified Application ID.
- int32 CFE_ES_GetAppInfo (CFE_ES_AppInfo_t *AppInfo, uint32 AppId)

  Get Application Information given a specified App ID.
- int32 CFE_ES_GetTaskInfo (CFE_ES_TaskInfo_t *TaskInfo, uint32 OSTaskId)

  Get Task Information given a specified Task ID.
- int32 CFE_ES_CreateChildTask (uint32 *TaskIdPtr, const char *TaskName, CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr, uint32 *StackPtr, uint32 StackSize, uint32 Priority, uint32 Flags)

  Creates a new task under an existing Application.
- int32 CFE_ES_RegisterChildTask (void)

  Registers a cFE Child task associated with a cFE Application.
- void CFE_ES_IncrementTaskCounter (void)

  Increments the execution counter for the calling task.
- int32 CFE_ES_DeleteChildTask (uint32 OSTaskId)

  Deletes a task under an existing Application.
- void CFE_ES_ExitChildTask (void)

  Exits a child task.
- int32 CFE_ES_WriteToSysLog (const char *SpecStringPtr,...)
- uint32 CFE_ES_CalculateCRC (const void *DataPtr, uint32 DataLength, uint32 InputCRC, uint32 TypeCRC)

  Calculate a CRC on a block of memory.
- int32 CFE_ES_RegisterCDS (CFE_ES_CDSHandle_t *CDSHandlePtr, int32 BlockSize, const char *Name)

Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)

- int32 CFE_ES_CopyToCDS (CFE_ES_CDSHandle_t Handle, void ∗DataToCopy)

  Save a block of data in the Critical Data Store (CDS)
- int32 CFE_ES_RestoreFromCDS (void ∗RestoreToMemory, CFE_ES_CDSHandle_t Handle)

  Recover a block of data from the Critical Data Store (CDS)
- int32 CFE_ES_RegisterGenCounter (uint32 ∗CounterIdPtr, const char ∗CounterName)

  Register a generic counter.
- int32 CFE_ES_DeleteGenCounter (uint32 CounterId)

  Delete a generic counter.
- int32 CFE_ES_IncrementGenCounter (uint32 CounterId)

  Increments the specified generic counter.
- int32 CFE_ES_SetGenCount (uint32 CounterId, uint32 Count)

  Set the specified generic counter.
- int32 CFE_ES_GetGenCount (uint32 CounterId, uint32 ∗Count)

  Get the specified generic counter count.
- int32 CFE_ES_GetGenCounterIDByName (uint32 ∗CounterIdPtr, const char ∗CounterName)

  Get the Id associated with a generic counter name.
- int32 CFE_ES_GetAppIDInternal (uint32 ∗AppIdPtr)
- void CFE_ES_LockSharedData (const char ∗FunctionName, int32 LineNumber)
- void CFE_ES_UnlockSharedData (const char ∗FunctionName, int32 LineNumber)
- void CFE_ES_ProcessCoreException (uint32 HostTaskId, const char ∗ReasonString, const uint32 ∗Context↩ Pointer, uint32 ContextSize)

  Process an exception detected by the underlying OS/PSP.

13.15.1 Function Documentation

13.15.1.1 CFE_ES_CalculateCRC()

```
uint32 CFE_ES_CalculateCRC (
            const void * DataPtr,
            uint32 DataLength,
            uint32 InputCRC,
            uint32 TypeCRC )
```

Description

This routine calculates a cyclic redundancy check (CRC) on a block of memory. The CRC algorithm used is determined by the last parameter.

Assumptions, External Events, and Notes:

None

Parameters

in	*DataPtr*	Pointer to the base of the memory block.
in	*DataLength*	The number of bytes in the memory block.
in	*InputCRC*	A starting value for use in the CRC calculation. This parameter allows the user to calculate the CRC of non-contiguous blocks as a single value. Nominally, the user should set this value to zero.
in	*TypeCRC*	One of the following CRC algorithm selections: • `CFE_MISSION_ES_CRC_8` - (Not currently implemented) • `CFE_MISSION_ES_CRC_16` - a CRC-16 algorithm • `CFE_MISSION_ES_CRC_32` - (not currently implemented)

Returns

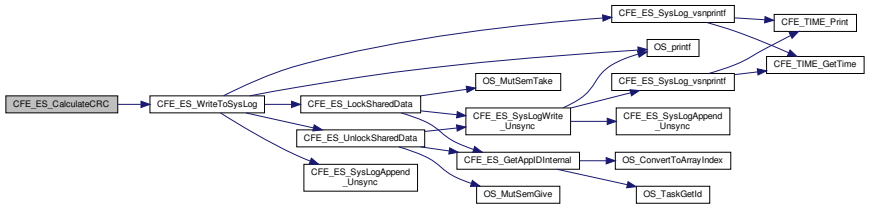
The result of the CRC calculation on the specified memory block.

Definition at line 1379 of file cfe_es_api.c.

References CFE_ES_WriteToSysLog(), CFE_MISSION_ES_CRC_16, CFE_MISSION_ES_CRC_32, and CFE_MISSION_ES_CRC_8.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), and CFE_ES_TaskInit().

Here is the call graph for this function:



13.15.1.2 CFE_ES_CopyToCDS()

```
int32 CFE_ES_CopyToCDS (
            CFE_ES_CDSHandle_t Handle,
            void * DataToCopy )
```

Description

This routine copies a specified block of memory into the Critical Data Store that had been previously registered via CFE_ES_RegisterCDS. The block of memory to be copied must be at least as big as the size specified when registering the CDS.

Assumptions, External Events, and Notes:

None

Parameters

in	*Handle*	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS.
in	*DataToCopy*	A Pointer to the block of memory to be copied into the CDS.

OS_SUCCESS	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.
OS_ERROR	Problem with handle or a size mismatch

Returns

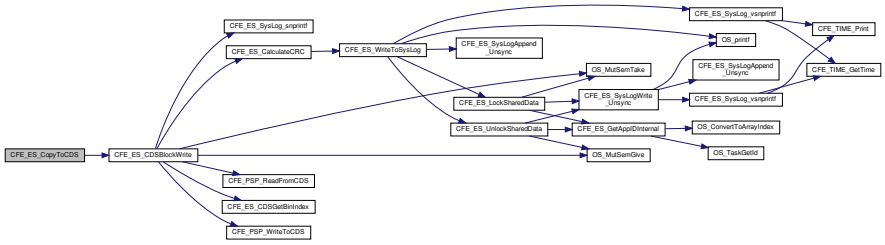
See also

CFE_ES_RegisterCDS, CFE_ES_RestoreFromCDS

Definition at line 1546 of file cfe_es_api.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDSBlockWrite(), CFE_ES_Global, CFE_ES_CDS_RegRec_t::↩ MemHandle, and CFE_ES_CDSVariables_t::Registry.

Here is the call graph for this function:



13.15.1.3 CFE_ES_CreateChildTask()

```
int32 CFE_ES_CreateChildTask (
            uint32 * TaskIdPtr,
            const char * TaskName,
            CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr,
            uint32 * StackPtr,
            uint32 StackSize,
            uint32 Priority,
            uint32 Flags )
```

Description

This routine creates a new task (a separate execution thread) owned by the calling Application.

Assumptions, External Events, and Notes:

None

Parameters

in	*TaskIdPtr*	A pointer to a variable that will be filled in with the new task's ID.
in	*TaskName*	A pointer to a string containing the desired name of the new task. This can be up to OS_MAX_API_NAME characters, including the trailing null.
in	*FunctionPtr*	A pointer to the function that will be spawned as a new task. This function must have the following signature: uint32 function(void). Input parameters for the new task are not supported.
in	*StackPtr*	A pointer to the location where the child task's stack pointer should start. NOTE: Not all underlying operating systems support this parameter.
in	*StackSize*	The number of bytes to allocate for the new task's stack.
in	*Priority*	The priority for the new task. Lower numbers are higher priority, with 0 being the highest priority. Applications cannot create tasks with a higher priority (lower number) than their own priority.
in	*Flags*	Reserved for future expansion.
out	∗*TaskIdPtr*	The Task ID of the newly created child task.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_CHILD_TASK_CREATE	There was an error creating a child task.

Returns

See also

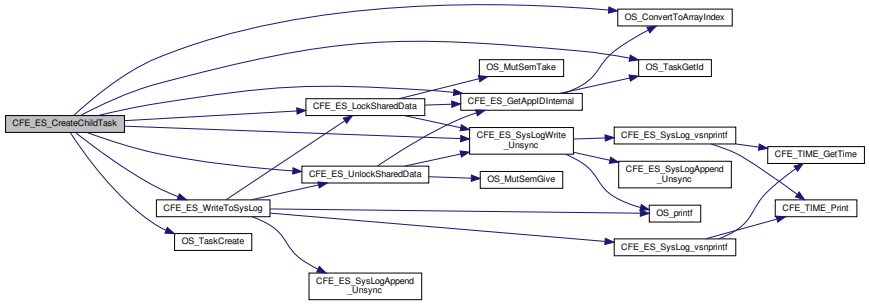
CFE_ES_RegisterChildTask, CFE_ES_DeleteChildTask, CFE_ES_ExitChildTask

Definition at line 985 of file cfe_es_api.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_Global_t::AppTable, CFE_ES_BAD_ARGUMENT, CFE_ES_ERR_CHILD_TASK_CREATE, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_SUCCESS, CFE_ES_MainTaskInfo_t::MainTaskId, NULL, OS_ConvertToArrayIndex(), OS_FP_ENABLED, OS_MAX_API_NAME, OS_SUCCESS, OS_TaskCreate(), OS_TaskGetId(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_TaskRecord_t::TaskName, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_StopPerfDataCmd().

Here is the call graph for this function:



13.15.1.4 CFE_ES_DeleteApp()

```
int32 CFE_ES_DeleteApp (
            uint32 AppID )
```

Description

This API causes a cFE Application to be stopped deleted.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppID*	Identifies the application to be reset.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

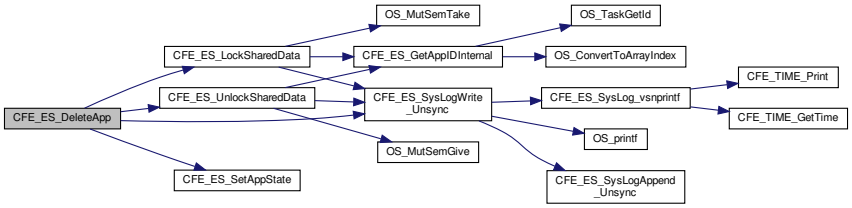
CFE_ES_RestartApp, CFE_ES_ReloadApp

Definition at line 333 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_RUNNING, CFE_ES_AppState_WAITING, CFE_ES_AppType_CORE, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_SYS_DELETE, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_APP_KILL_TIMEOUT, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::Name, CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_StopAppCmd().

Here is the call graph for this function:



13.15.1.5 CFE_ES_DeleteChildTask()

```
int32 CFE_ES_DeleteChildTask (
            uint32 TaskId )
```

Description

This routine deletes a task under an Application specified by the `TaskId` obtained when the child task was created using the CFE_ES_CreateChildTask API.

Assumptions, External Events, and Notes:

None

Parameters

in	*TaskId*	The task ID previously obtained when the Child Task was created with the CFE_ES_CreateChildTask API.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

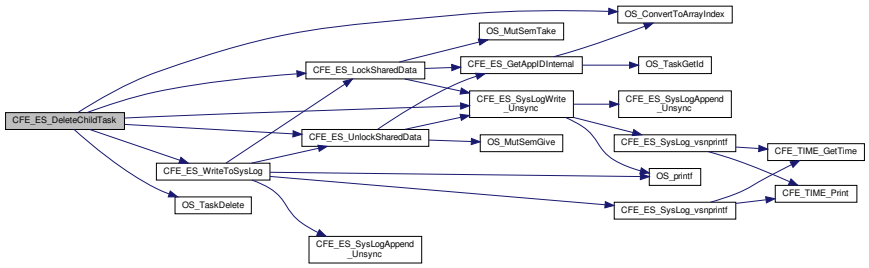
See also

CFE_ES_RegisterChildTask, CFE_ES_CreateChildTask, CFE_ES_ExitChildTask

Definition at line 1172 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_CHILD_TASK_DELETE, CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK, CFE_ES_ERR_TASKID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_MainTaskInfo_t::MainTaskId, OS_ConvertToArrayIndex(), OS_MAX_TASKS, OS_SUCCESS, OS_TaskDelete(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppRecord_t::TaskInfo, and CFE_ES_Global_t::TaskTable.

Here is the call graph for this function:



13.15.1.6 CFE_ES_DeleteGenCounter()

```
int32 CFE_ES_DeleteGenCounter (
            uint32 CounterId )
```

Description

This routine deletes a previously registered generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter Id of the newly created counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_IncrementGenCounter, CFE_ES_RegisterGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

Definition at line 1612 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.15.1.7 CFE_ES_ExitApp()

```
void CFE_ES_ExitApp (
            uint32 ExitStatus )
```

Description

This API is the "Exit Point" for the cFE application

Assumptions, External Events, and Notes:

None

Parameters

in	*ExitStatus*	.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

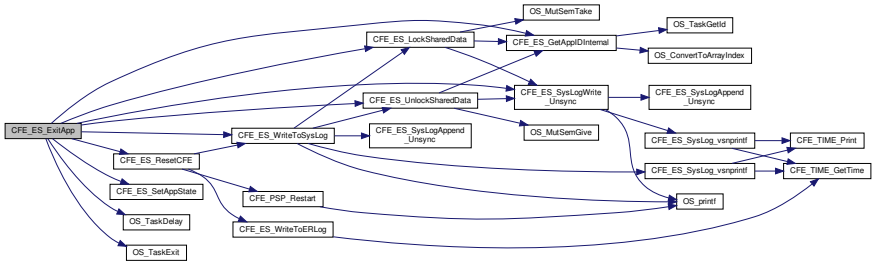
CFE_ES_RunLoop, CFE_ES_RegisterApp

Definition at line 375 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_Global_t::AppTable, CFE_ES_AppState_STOPPED, CFE_ES_AppType_CORE, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_ResetCFE(), CFE_ES_RunStatus_APP_ERROR, CFE_ES_RunStatus_APP_EXIT, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::Name, OS_TaskDelay(), OS_TaskExit(), CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.15.1.8 CFE_ES_ExitChildTask()

```
void CFE_ES_ExitChildTask (
             void  )
```

Description

This routine allows the current executing child task to exit and be deleted by ES.

Assumptions, External Events, and Notes:

This function cannot be called from an Application's Main Task.

This function does not return a value, but if it does return at all, it is assumed that the Task was either unregistered or this function was called from a cFE Application's main task.

Returns

See also

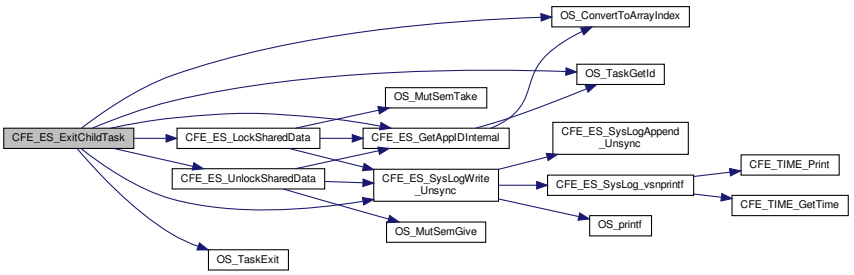
CFE_ES_RegisterChildTask, CFE_ES_CreateChildTask, CFE_ES_DeleteChildTask

Definition at line 1281 of file cfe_es_api.c.

References CFE_ES_Global_t::AppTable, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_SUCCESS, CFE_ES_MainTaskInfo_t::MainTaskId, OS_ConvertToArrayIndex(), OS_SUCCESS, OS_TaskExit(), OS_TaskGetId(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppRecord_t::TaskInfo, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_PerfLogDump().

Here is the call graph for this function:



13.15.1.9 CFE_ES_GetAppID()

```
int32 CFE_ES_GetAppID (
            uint32 * AppIdPtr )
```

Description

This routine retrieves the cFE Application ID for the calling Application.

Assumptions, External Events, and Notes:

NOTE: **All** tasks associated with the Application would return the same Application ID.

Parameters

in	*AppIdPtr*	Pointer to variable that is to receive the Application's ID.
out	∗*AppIdPtr*	Application ID of the calling Application.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

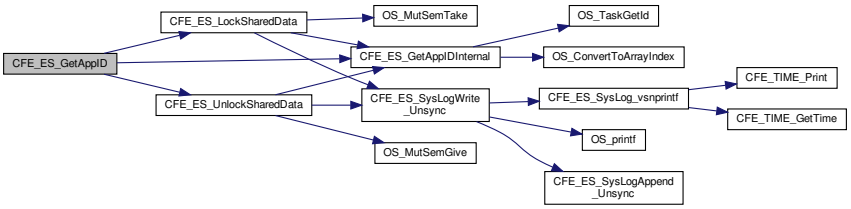
CFE_ES_GetResetType, CFE_ES_GetAppIDByName, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

Definition at line 801 of file cfe_es_api.c.

References CFE_ES_GetAppIDInternal(), CFE_ES_LockSharedData(), and CFE_ES_UnlockSharedData().

Referenced by CFE_ES_CDS_ValidateAppID(), CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_SB_↩ AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipe(), CFE_SB_GetLastSenderId(), CFE_SB_LockSharedData(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnlockSharedData(), CFE_↩ SB_Unsubscribe(), CFE_SB_UnsubscribeLocal(), CFE_SB_ZeroCopyGetPtr(), and EVS_GetAppID().

Here is the call graph for this function:



13.15.1.10 CFE_ES_GetAppIDByName()

```
int32 CFE_ES_GetAppIDByName (
            uint32 * AppIdPtr,
            const char * AppName )
```

Description

This routine retrieves the cFE Application ID associated with a specified Application name.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppIdPtr*	Pointer to variable that is to receive the Application's ID.
in	*AppName*	Pointer to null terminated character string containing an Application name.
out	∗*AppIdPtr*	Application ID of the calling Application.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

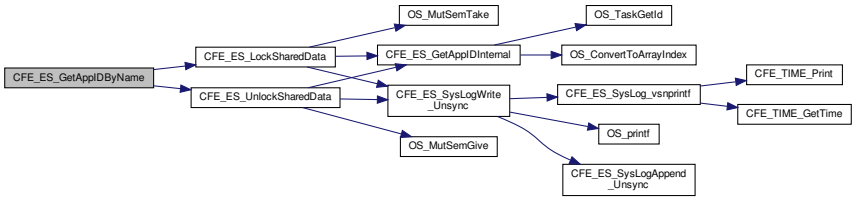
CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

Definition at line 765 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CF↩E_ES_ERR_APPNAME, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_PLAT↩FORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppStartParams_t::Name, OS_MAX_API_NAME, and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_Restart↩AppCmd(), CFE_ES_StopAppCmd(), and EVS_GetApplicationInfo().

Here is the call graph for this function:



13.15.1.11 CFE_ES_GetAppIDInternal()

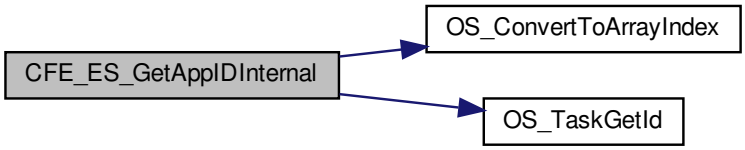
```
int32 CFE_ES_GetAppIDInternal (
            uint32 * AppIdPtr )
```

Definition at line 1728 of file cfe_es_api.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_SUCCESS, OS_Convert↩ ToArrayIndex(), OS_SUCCESS, OS_TaskGetId(), CFE_ES_TaskRecord_t::RecordUsed, and CFE_ES_Global_t::↩ TaskTable.

Referenced by CFE_ES_CreateChildTask(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_GetAppID(), C↩ FE_ES_LockSharedData(), CFE_ES_RunLoop(), CFE_ES_UnlockSharedData(), and CFE_ES_WaitForSystemState().

Here is the call graph for this function:



13.15.1.12 CFE_ES_GetAppInfo()

```
int32 CFE_ES_GetAppInfo (
            CFE_ES_AppInfo_t * AppInfo,
            uint32 AppId )
```

Description

This routine retrieves the information about an App associated with a specified App ID. The information includes all of the information ES maintains for an application (documented in the CFE_ES_AppInfo_t type)

Assumptions, External Events, and Notes:

None

Parameters

in	*AppInfo*	Pointer to a CFE_ES_AppInfo_t structure that holds the specific Application information.
in	*AppId*	Application ID of Application whose name is being requested.
out	**AppInfo*	Filled out CFE_ES_AppInfo_t structure containing the App Name, and application memory addresses among other fields.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

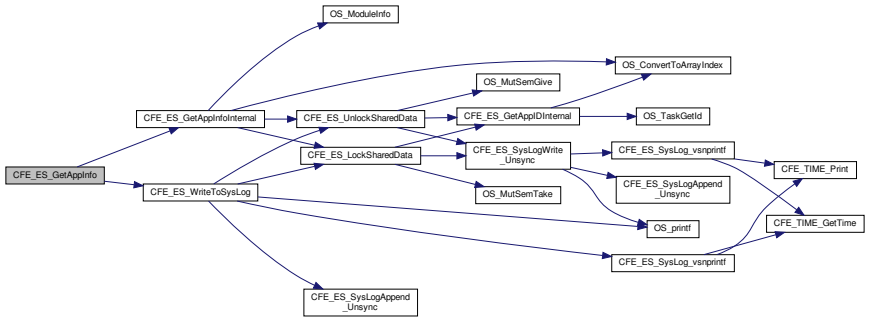
See also

CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName

Definition at line 872 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_APPID, CFE_ES_ERR_BUFFER, CFE_ES_GetAppInfoInternal(), CFE_ES_Global, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX_APPLICATIONS, and CFE_SUCCESS.

Here is the call graph for this function:



13.15.1.13 CFE_ES_GetAppName()

```
int32 CFE_ES_GetAppName (
            char * AppName,
            uint32 AppId,
            uint32 BufferLength )
```

Description

This routine retrieves the cFE Application name associated with a specified Application ID.

Assumptions, External Events, and Notes:

In the case of a failure (CFE_ES_ERR_APPID), an empty string is returned. CFE_ES_ERR_APPID will be returned if the specified Application ID (AppId) is invalid or not in use.

Parameters

in	*AppName*	Pointer to a character array of at least `BufferLength` in size that will be filled with the appropriate Application name.
in	*AppId*	Application ID of Application whose name is being requested.
in	*BufferLength*	The maximum number of characters, including the null terminator, that can be put into the `AppName` buffer. This routine will truncate the name to this length, if necessary.
out	∗*AppName*	Null terminated Application name of the Application associated with the specified Application ID.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.

Returns

See also

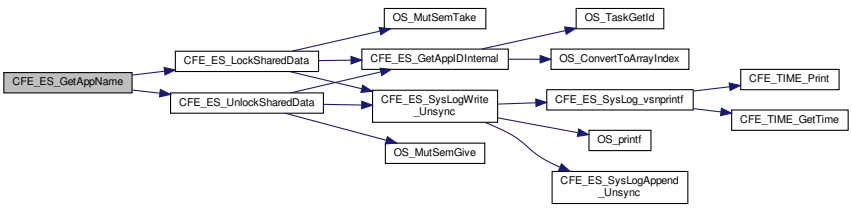
CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetTaskInfo

Definition at line 822 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CF↩E_ES_ERR_APPID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_PLATFOR↩M_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppStartParams_t::Name, and CFE_ES_AppRecord_t::↩StartParams.

Referenced by CFE_ES_FormCDSName(), CFE_ES_RegisterCDS(), CFE_EVS_WriteAppDataFileCmd(), CFE_SB↩_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_SendRtgInfo(), CFE_SB_SetPipeOpts(), EVS_GenerateEvent↩Telemetry(), EVS_IsFiltered(), and EVS_NotRegistered().

Here is the call graph for this function:



13.15.1.14 CFE_ES_GetGenCount()

```
int32 CFE_ES_GetGenCount (
            uint32 CounterId,
            uint32 * Count )
```

Description

This routine gets the value of a generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter to get the value from.
in	*Count	The value of the Counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_IncrementGenCounter, CFE_ES_GetGenCounterIDByName

Definition at line 1673 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, NULL, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.15.1.15 CFE_ES_GetGenCounterIDByName()

```
int32 CFE_ES_GetGenCounterIDByName (
            uint32 * CounterIdPtr,
            const char * CounterName )
```

Description

This routine gets the Counter Id for a generic counter specified by name.

Assumptions, External Events, and Notes:

None.

Parameters

in	*∗CounterName*	The name of the Counter.
out	*∗CounterIdPtr*	The Counter Id for the given name.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_Increment↩ GenCounter, CFE_ES_GetGenCount

Definition at line 1687 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_↩ SUCCESS, CFE_ES_GenCounterRecord_t::CounterName, CFE_ES_Global_t::CounterTable, NULL, OS_MAX_API↩ _NAME, and CFE_ES_GenCounterRecord_t::RecordUsed.

Referenced by CFE_ES_RegisterGenCounter().

13.15.1.16 CFE_ES_GetResetType()

```
int32 CFE_ES_GetResetType (
            uint32 * ResetSubtypePtr )
```

Description

Provides the caller with codes that identifies the type of Reset the processor most recently underwent. The caller can also obtain information on what caused the reset by supplying a pointer to a variable that will be filled with the Reset Sub-Type.

Assumptions, External Events, and Notes:

None

Parameters

in	*ResetSubtypePtr*	Pointer to uint32 type variable in which the Reset Sub-Type will be stored. The caller can set this pointer to NULL if the Sub-Type is of no interest.
out	*∗ResetSubtypePtr*	If the provided pointer was not NULL, the Reset Sub-Type is stored at the given address. For a list of possible Sub-Type values, see "Reset Sub-Types" .

CFE_PSP_RST_TYPE_POWERON	All memory has been cleared
CFE_PSP_RST_TYPE_PROCESSOR	Volatile disk, Critical Data Store and User Reserved memory could still be valid

Returns

See also

CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

Definition at line 64 of file cfe_es_api.c.

References CFE_ES_ResetDataPtr, NULL, CFE_ES_ResetVariables_t::ResetSubtype, CFE_ES_ResetVariables_t::↩ ResetType, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_EVS_EarlyInit().

13.15.1.17 CFE_ES_GetTaskInfo()

```
int32 CFE_ES_GetTaskInfo (
            CFE_ES_TaskInfo_t * TaskInfo,
            uint32 TaskId )
```

Description

This routine retrieves the information about a Task associated with a specified Task ID. The information includes Task Name, and Parent/Creator Application ID.

Assumptions, External Events, and Notes:

None

Parameters

in	*TaskInfo*	Pointer to a CFE_ES_TaskInfo_t structure that holds the specific task information.
in	*TaskId*	Application ID of Application whose name is being requested.
out	∗*TaskInfo*	Filled out CFE_ES_TaskInfo_t structure containing the Task Name, Parent App Name, Parent App ID among other fields.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_TASKID	Occurs when the Task ID passed into CFE_ES_GetTaskInfo is invalid.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

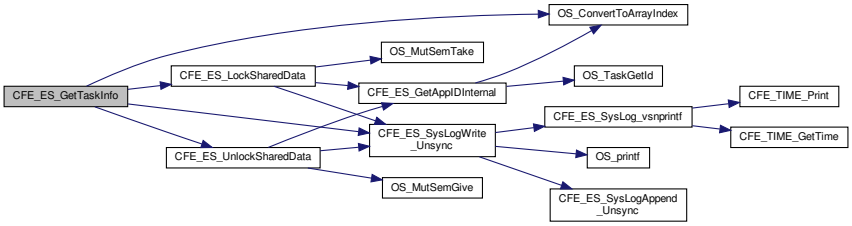
CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName

Definition at line 912 of file cfe_es_api.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_TaskInfo_t::AppId, CFE_ES_TaskInfo_t::AppName, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_TASKID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_SUCCESS, CFE_ES_TaskRecord_t::ExecutionCounter, CFE_ES_TaskInfo_t::ExecutionCounter, CFE_ES_AppStartParams_t::Name, OS_ConvertToArrayIndex(), OS_MAX_API_NAME, OS_MAX_TASKS, OS_SUCCESS, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskInfo_t::TaskId, CFE_ES_TaskRecord_t::TaskName, CFE_ES_TaskInfo_t::TaskName, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_ListTasks(), CFE_ES_ProcessCoreException(), CFE_ES_QueryAllTasksCmd(), and CFE_SB_GetAppTskName().

Here is the call graph for this function:



13.15.1.18 CFE_ES_IncrementGenCounter()

```
int32 CFE_ES_IncrementGenCounter (
            uint32 CounterId )
```

Description

This routine increments the specified generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter to be incremented.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

Definition at line 1634 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.15.1.19 CFE_ES_IncrementTaskCounter()

```
void CFE_ES_IncrementTaskCounter (
            void  )
```

Description

This routine increments the execution counter that is stored for the calling task. It can be called from cFE Application main tasks, child tasks, or cFE Core application main tasks. Normally, the call is not necessary from a cFE Application, since the CFE_ES_RunLoop call increments the counter for the Application.

Assumptions, External Events, and Notes:

NOTE: This API is not needed for Appplications that call the CFE_ES_RunLoop call.

This function does not return a value.

Returns

See also

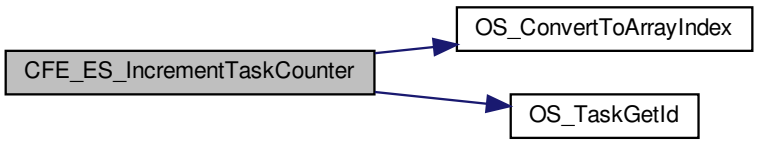
CFE_ES_RunLoop

Definition at line 1153 of file cfe_es_api.c.

References CFE_ES_Global, CFE_ES_TaskRecord_t::ExecutionCounter, OS_ConvertToArrayIndex(), OS_SUCCESS, OS_TaskGetId(), and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.15.1.20 CFE_ES_LockSharedData()

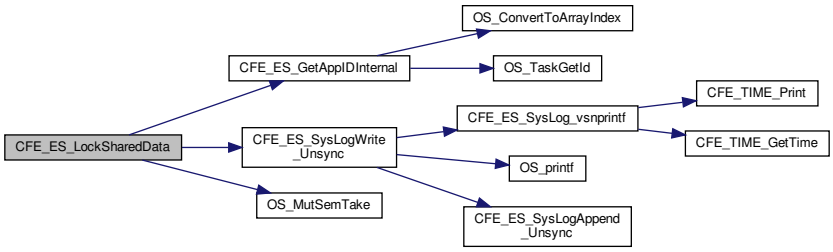
```
void CFE_ES_LockSharedData (
            const char * FunctionName,
            int32 LineNumber )
```

Definition at line 1775 of file cfe_es_api.c.

References CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_SysLogWrite_Unsync(), OS_MutSemTake(), OS_SUCCESS, and CFE_ES_Global_t::SharedDataMutex.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_ClearSyslogCmd(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_GetAppID(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_ES_LoadLibrary(), CFE_ES_MainTaskSyncDelay(), CFE_ES_RegisterApp(), CFE_ES_RegisterChildTask(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_SysLogDump(), CFE_ES_WaitForSystemState(), and CFE_ES_WriteToSysLog().

Here is the call graph for this function:



13.15.1.21 CFE_ES_ProcessCoreException()

```
void CFE_ES_ProcessCoreException (
            uint32 HostTaskId,
            const char * ReasonString,
            const uint32 * ContextPointer,
            uint32 ContextSize )
```

Description

This hook routine is called from the PSP when an exception occurs

Assumptions, External Events, and Notes:

None.

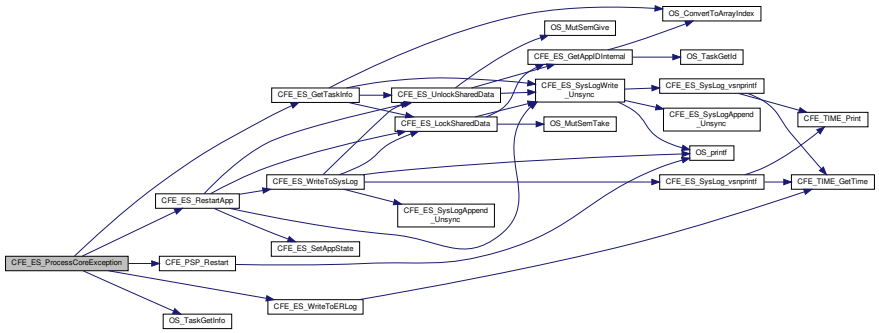
Parameters

in	*HostTaskId*	The OS (not OSAL) task ID
in	*ReasonString*	Identifier from PSP
in	*ContextPointer*	Context data from PSP
in	*ContextSize*	Size of context data from PSP

Definition at line 1852 of file cfe_es_api.c.

References CFE_ES_TaskInfo_t::AppId, CFE_ES_Global_t::AppTable, CFE_ES_APP_RESTART, CFE_ES_↩ ExceptionAction_RESTART_APP, CFE_ES_GetTaskInfo(), CFE_ES_Global, CFE_ES_LogEntryType_CORE, CF↩ E_ES_ResetDataPtr, CFE_ES_RestartApp(), CFE_ES_WriteToERLog(), CFE_PSP_Restart(), CFE_PSP_RST_SU↩ BTYPE_EXCEPTION, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_SUCCE↩ SS, CFE_ES_ResetVariables_t::ES_CausedReset, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_Reset↩ Variables_t::MaxProcessorResetCount, OS_MAX_TASKS, OS_SUCCESS, OS_TaskGetInfo(), OS_task_prop_t::O↩ Stask_id, CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Reset↩ Data_t::ResetVars, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, and CFE_ES_Global_t::↩ TaskTable.

Here is the call graph for this function:



13.15.1.22 CFE_ES_RegisterApp()

```
int32 CFE_ES_RegisterApp (
            void  )
```

Description

This API registers the calling Application with the cFE.

Assumptions, External Events, and Notes:

NOTE: This function **MUST** be called before any other cFE API functions are called.

Return codes from OS_TaskRegister	
Return codes from OS_BinSemTake	
CFE_SUCCESS	Operation was performed successfully

Returns

See also

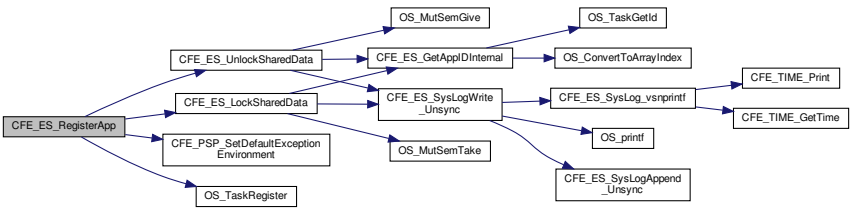
CFE_ES_ExitApp, CFE_ES_RunLoop

Definition at line 722 of file cfe_es_api.c.

References CFE_ES_ERR_APP_REGISTER, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_P↩SP_SetDefaultExceptionEnvironment(), CFE_SUCCESS, OS_SUCCESS, and OS_TaskRegister().

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), and CFE_SB_AppInit().

Here is the call graph for this function:



13.15.1.23 CFE_ES_RegisterCDS()

```
int32 CFE_ES_RegisterCDS (
            CFE_ES_CDSHandle_t * HandlePtr,
            int32 BlockSize,
            const char * Name )
```

Description

This routine allocates a block of memory in the Critical Data Store and associates it with the calling Application. The memory can survive an Application restart as well as a Processor Reset.

Assumptions, External Events, and Notes:

None

Parameters

in	*HandlePtr*	Pointer Application's variable that will contain the CDS Memory Block Handle.
in	*BlockSize*	The number of bytes needed in the CDS.
in	*Name*	A pointer to a character string containing an application unique name of CFE_MISSION_ES_CDS_MAX_NAME_LENGTH characters or less.
out	*∗HandlePtr*	The handle of the CDS block that can be used in CFE_ES_CopyToCDS and CFE_ES_RestoreFromCDS.

CFE_SUCCESS	The memory block was successfully created in the CDS.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.
CFE_ES_CDS_ALREADY_EXISTS	The Application is receiving the pointer to a CDS that was already present.
CFE_ES_CDS_INVALID_SIZE	The Application is requesting a CDS Block with a size of zero.
CFE_ES_CDS_INVALID_NAME	The Application is requesting a CDS Block with an invalid ASCII string name. Either the name is too long (> CFE_MISSION_ES_CDS_MA↩ X_NAME_LENGTH) or was an empty string.
CFE_ES_CDS_REGISTRY_FULL	The CDS Registry has as many entries in it as it can hold. The CDS Registry size can be adjusted with the CFE_PLATFORM_ES_CDS_↩ MAX_NUM_ENTRIES macro defined in the cfe_platform_cfg.h file.

Returns

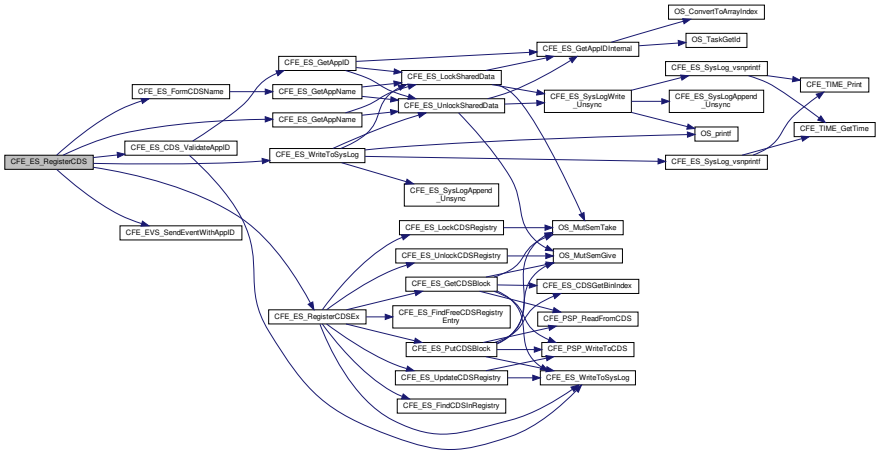
See also

CFE_ES_CopyToCDS, CFE_ES_RestoreFromCDS

Definition at line 1466 of file cfe_es_api.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_BAD_HANDLE, CFE_ES_CDS_INVALID_NAME, CFE_ES_CDS_INVALID_SIZE, CFE_ES_CDS_MAX_FULL_NAME_LEN, CFE_ES_CDS_REGISTER_ERR_EID, CFE_ES_CDS_ValidateAppID(), CFE_ES_FormCDSName(), CFE_ES_GetAppName(), CFE_ES_Global, CFE_ES_NOT_IMPLEMENTED, CFE_ES_RegisterCDSEx(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_MISSION_ES_CDS_MAX_NAME_LENGTH, CFE_SUCCESS, CFE_ES_CDS_Variables_t::MemPoolSize, and OS_MAX_API_NAME.

Here is the call graph for this function:



13.15.1.24 CFE_ES_RegisterChildTask()

```
int32 CFE_ES_RegisterChildTask (
            void  )
```

Description

This routine registers a cFE Child task and associates it with its parent cFE Application.

Assumptions, External Events, and Notes:

NOTE: This API **MUST** be called by the Child Task before any other cFE API calls are made.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_CHILD_TASK_REGISTER	Errors occured when trying to register a child task.

Returns

See also

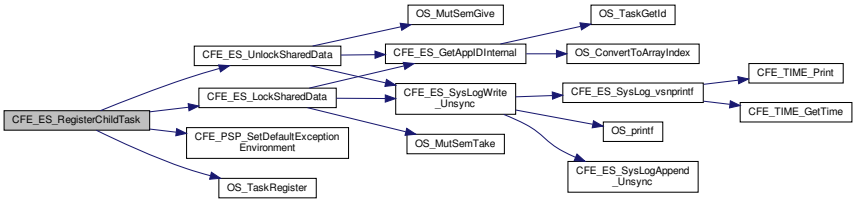
CFE_ES_CreateChildTask, CFE_ES_DeleteChildTask, CFE_ES_ExitChildTask

Definition at line 1111 of file cfe_es_api.c.

References CFE_ES_ERR_CHILD_TASK_REGISTER, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_PSP_SetDefaultExceptionEnvironment(), CFE_SUCCESS, OS_SUCCESS, and OS_TaskRegister().

Referenced by CFE_ES_PerfLogDump().

Here is the call graph for this function:



13.15.1.25 CFE_ES_RegisterGenCounter()

```
int32 CFE_ES_RegisterGenCounter (
            uint32 * CounterIdPtr,
            const char * CounterName )
```

Description

This routine registers a generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterName	The Name of the generic counter.
out	*CounterIdPtr	The Counter Id of the newly created counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

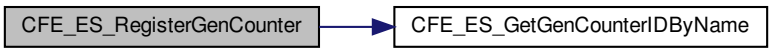
See also

CFE_ES_IncrementGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGen↩ Count, CFE_ES_GetGenCounterIDByName

Definition at line 1573 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_GetGenCounterIDByName(), CFE_ES_Global, CFE_PLAT↩ FORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Gen↩ CounterRecord_t::CounterName, CFE_ES_Global_t::CounterTable, NULL, OS_MAX_API_NAME, and CFE_ES_↩ GenCounterRecord_t::RecordUsed.

Here is the call graph for this function:



13.15.1.26 CFE_ES_ReloadApp()

```
int32 CFE_ES_ReloadApp (
            uint32 AppID,
            const char * AppFileName )
```

Description

This API causes a cFE Application to be stopped and restarted from the specified file.

Assumptions, External Events, and Notes:

The specified application will be deleted before it is reloaded from the specified file. In the event that an application cannot be reloaded due to a corrupt file, the application may no longer be reloaded when given a valid load file (it has been deleted and no longer exists). To recover, the application may be restarted by loading the application via the ES_STARTAPP command (CFE_ES_START_APP_CC).

Parameters

in	*AppID*	Identifies the application to be reset.
in	*AppFileName*	Identifies the new file to start.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

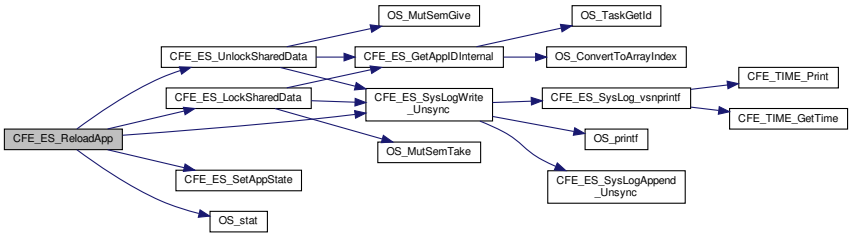
CFE_ES_RestartApp, CFE_ES_DeleteApp, CFE_ES_START_APP_CC

Definition at line 276 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_RUNNING, CFE_ES_AppState_WAITING, CFE_ES_AppType_CORE, CFE_ES_ERR_APPID, CFE_ES_FILE_IO_ERR, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_SYS_RELOAD, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_APP_KILL_TIMEOUT, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::FileName, CFE_ES_AppStartParams_t::Name, OS_MAX_PATH_LEN, OS_stat(), OS_SUCCESS, CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ReloadAppCmd().

Here is the call graph for this function:



13.15.1.27 CFE_ES_ResetCFE()

```
int32 CFE_ES_ResetCFE (
            uint32 ResetType )
```

Description

This API causes an immediate reset of the cFE Kernel and all cFE Applications. The caller can specify whether the reset should clear all memory (CFE_PSP_RST_TYPE_POWERON) or try to retain volatile memory areas (CFE_PSP_RST_TYPE_PROCESSOR).

Assumptions, External Events, and Notes:

None

Parameters

in	*ResetType*	Identifies the type of reset desired. Allowable settings are: • CFE_PSP_RST_TYPE_POWERON - Causes all memory to be cleared • CFE_PSP_RST_TYPE_PROCESSOR - Attempts to retain volatile disk, critical data store and user reserved memory.

CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.
CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

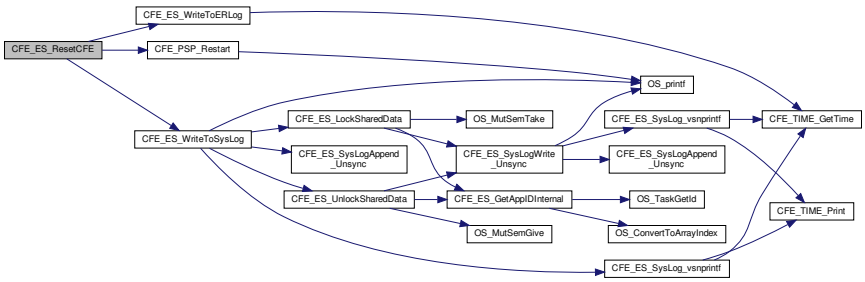
CFE_ES_Main

Definition at line 82 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_LogEntryType_CORE, CFE_ES_NOT_IMPLEMENTED, CFE_ES_ResetDataPtr, CFE_ES_WriteToERLog(), CFE_ES_WriteToSysLog(), CFE_PSP_Restart(), CFE_PSP_RST_SUBTYPE_RESET_COMMAND, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_ES_ResetVariables_t::ES_CausedReset, CFE_ES_ResetVariables_t::MaxProcessorResetCount, NULL, CFE_ES_ResetVariables_t::ProcessorResetCount, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_ExitApp(), and CFE_ES_RestartCmd().

Here is the call graph for this function:



13.15.1.28 CFE_ES_RestartApp()

```
int32 CFE_ES_RestartApp (
            uint32 AppID )
```

Description

This API causes a cFE Application to be stopped and restarted.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppID*	Identifies the application to be reset.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

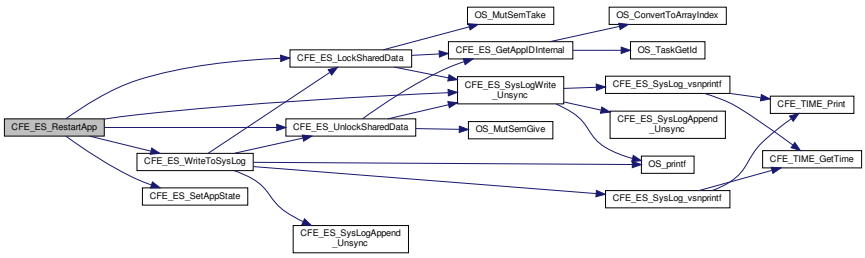
CFE_ES_ReloadApp, CFE_ES_DeleteApp

Definition at line 222 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_RUNNING, CFE_ES_AppState_WAITING, CFE_ES_AppType_CORE, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_SYS_RESTART, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_APP_KILL_TIMEOUT, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::Name, CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ProcessCoreException(), and CFE_ES_RestartAppCmd().

Here is the call graph for this function:



13.15.1.29 CFE_ES_RestoreFromCDS()

```
int32 CFE_ES_RestoreFromCDS (
            void * RestoreToMemory,
            CFE_ES_CDSHandle_t Handle )
```

Description

This routine copies data from the Critical Data Store identified with the `Handle` into the area of memory pointed to by the `RestoreToMemory` pointer. The area of memory to be copied into must be at least as big as the size specified when registering the CDS. The recovery will indicate an error if the data integrity check maintained by the CDS indicates the contents of the CDS have changed. However, the contents will still be copied into the specified area of memory.

Assumptions, External Events, and Notes:

None

Parameters

in	*Handle*	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS.
in	*RestoreToMemory*	A Pointer to the block of memory that is to be restored with the contents of the CDS.
out	∗*RestoreToMemory*	The contents of the specified CDS.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_CDS_BLOCK_CRC_ERR	Occurs when trying to read a CDS Data block and the CRC of the current data does not match the stored CRC for the data. Either the contents of the CDS Data Block are corrupted or the CDS Control Block is corrupted.
OS_ERROR	Problem with handle or a size mismatch

Returns

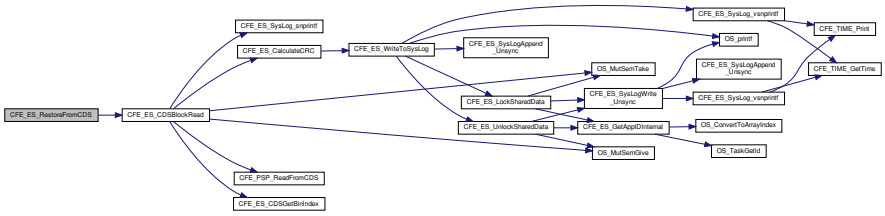
See also

CFE_ES_RegisterCDS, CFE_ES_CopyToCDS

Definition at line 1561 of file cfe_es_api.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDSBlockRead(), CFE_ES_Global, CFE_ES_CDS_RegRec_t::↩ MemHandle, and CFE_ES_CDSVariables_t::Registry.

Here is the call graph for this function:



13.15.1.30 CFE_ES_RunLoop()

```
bool CFE_ES_RunLoop (
            uint32 * ExitStatus )
```

Description

This is the API that allows an app to check for exit requests from the system.

Assumptions, External Events, and Notes:

None

Parameters

in	*ExitStatus*	A pointer to a variable containing the Application's desired run status. Acceptable values are: • CFE_ES_RunStatus_APP_RUN - Indicates that the Application should continue to run. • CFE_ES_RunStatus_APP_EXIT - Indicates that the Application wants to exit normally. • CFE_ES_RunStatus_APP_ERROR - Indicates that the Application is quitting with an error.

true	The application should continue executing
false	The application should terminate itself

Returns

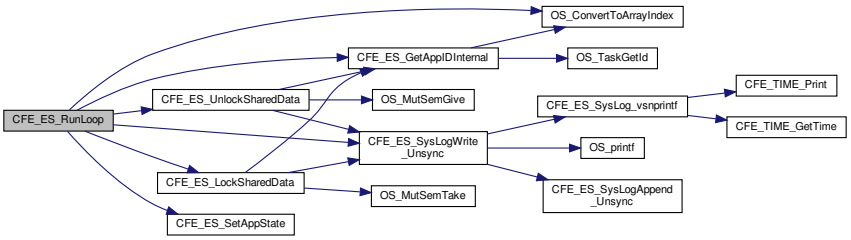
See also

CFE_ES_ExitApp, CFE_ES_RegisterApp

Definition at line 503 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_Global_t::AppTable, CFE_ES_AppState_RUNNI↩ NG, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_APP_ERR↩ OR, CFE_ES_RunStatus_APP_EXIT, CFE_ES_RunStatus_APP_RUN, CFE_ES_SetAppState(), CFE_ES_SysLog↩ Write_Unsync(), CFE_ES_UnlockSharedData(), CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_↩ TaskRecord_t::ExecutionCounter, CFE_ES_MainTaskInfo_t::MainTaskId, OS_ConvertToArrayIndex(), CFE_ES_App↩ Record_t::TaskInfo, and CFE_ES_Global_t::TaskTable.

Here is the call graph for this function:



13.15.1.31 CFE_ES_SetAppState()

```
void CFE_ES_SetAppState (
            uint32 AppID,
            uint32 TargetState )
```

Definition at line 191 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_MAX, CFE_ES_↩ AppState_UNDEFINED, and CFE_ES_Global.

Referenced by CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES↩ _RunLoop(), and CFE_ES_WaitForSystemState().

13.15.1.32 CFE_ES_SetGenCount()

```
int32 CFE_ES_SetGenCount (
            uint32 CounterId,
            uint32 Count )
```

Description

This routine sets the specified generic counter to the specified value.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter to be set.
in	*Count*	The new value of the Counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_IncrementGenCounter, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

Definition at line 1654 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.15.1.33 CFE_ES_UnlockSharedData()

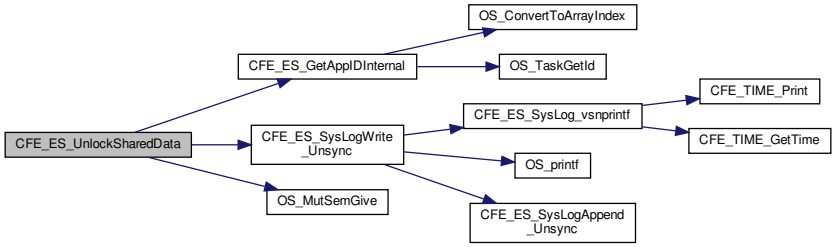
```
void CFE_ES_UnlockSharedData (
            const char * FunctionName,
            int32 LineNumber )
```

Definition at line 1813 of file cfe_es_api.c.

References CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_SysLogWrite_Unsync(), OS_MutSemGive(), OS_SUCCESS, and CFE_ES_Global_t::SharedDataMutex.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_ClearSyslogCmd(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_GetAppID(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_ES_LoadLibrary(), CFE_ES_MainTaskSyncDelay(), CFE_ES_RegisterApp(), CFE_ES_RegisterChildTask(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_SysLogDump(), CFE_ES_WaitForSystemState(), and CFE_ES_WriteToSysLog().

Here is the call graph for this function:



13.15.1.34 CFE_ES_WaitForStartupSync()

```
void CFE_ES_WaitForStartupSync (
            uint32 TimeOutMilliseconds )
```

Description

This is the API that allows an app to wait for the rest of the apps to complete their entire initialization before continuing. It is most useful for applications such as Health and Safety or the Scheduler that need to wait until applications exist and are running before sending out packets to them.

This is a specialized wrapper for CFE_ES_WaitForSystemState() for compatibility with applications using this API.

Assumptions, External Events, and Notes:

This API should only be called as the last item of an Apps initialization. In addition, this API should only be called by an App that is started from the ES Startup file. It should not be used by an App that is started after the system is running. (Although it will cause no harm)

Parameters

in	*TimeOutMilliseconds*	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.

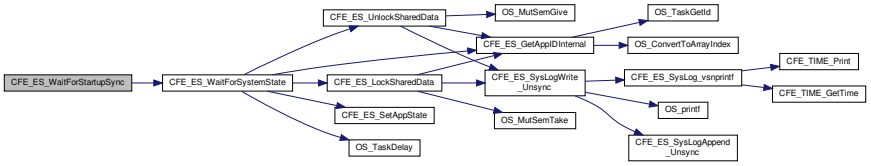
See also

CFE_ES_RunLoop

Definition at line 709 of file cfe_es_api.c.

References CFE_ES_SystemState_OPERATIONAL, and CFE_ES_WaitForSystemState().

Here is the call graph for this function:



13.15.1.35 CFE_ES_WaitForSystemState()

```
int32 CFE_ES_WaitForSystemState (
            uint32 MinSystemState,
            uint32 TimeOutMilliseconds )
```

Description

This is the API that allows an app to wait for the rest of the apps to complete a given stage of initialization before continuing.

This gives finer grained control than the "CFE_ES_WaitForStartupSync()" call.

Assumptions, External Events, and Notes:

This API assumes that the caller has also been initialized sufficiently to satisfy the global system state it is waiting for, and the apps own state will be updated accordingly.

Parameters

in	*TimeOutMilliseconds*	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.
in	*MinSystemState*	Determine the state of the App

Returns

if state was successfully achieved CFE_ES_OPERATION_TIMED_OUT if the timeout was reached (or other defined error code in case of error)

See also

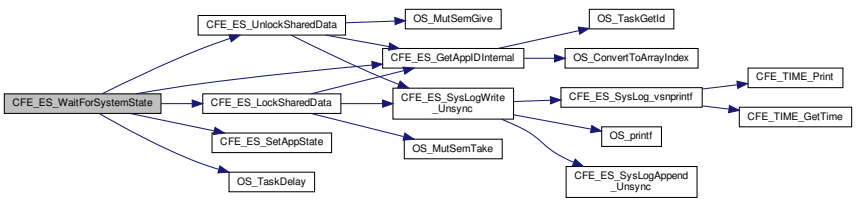
CFE_ES_RunLoop

Definition at line 607 of file cfe_es_api.c.

References CFE_ES_Global_t::AppTable, CFE_ES_AppState_EARLY_INIT, CFE_ES_AppState_LATE_INIT, CFE_ES_AppState_RUNNING, CFE_ES_AppState_STOPPED, CFE_ES_AppType_CORE, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_OPERATION_TIMED_OUT, CFE_ES_SetAppState(), CFE_ES_SystemState_APPS_INIT, CFE_ES_SystemState_CORE_READY, CFE_ES_SystemState_OPERATIONAL, CFE_ES_SystemState_SHUTDOWN, CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC, CFE_SUCCESS, OS_TaskDelay(), CFE_ES_Global_t::SystemState, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_TaskMain(), CFE_ES_WaitForStartupSync(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.15.1.36 CFE_ES_WriteToSysLog()

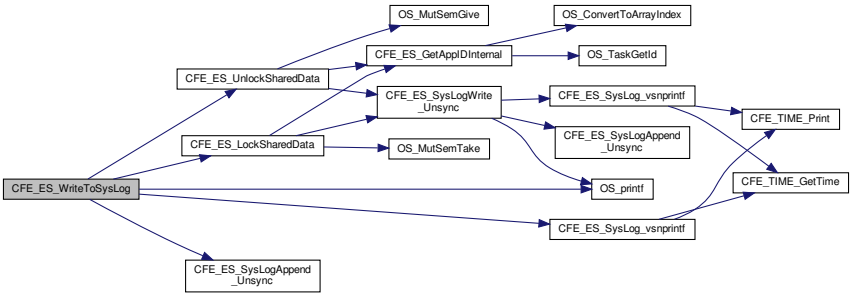
```
int32 CFE_ES_WriteToSysLog (
            const char * SpecStringPtr,
             ...  )
```

Definition at line 1347 of file cfe_es_api.c.

References CFE_ES_LockSharedData(), CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_SysLog_vsnprintf(), CFE↩_ES_SysLogAppend_Unsync(), CFE_ES_UnlockSharedData(), and OS_printf().

Referenced by CFE_ES_CalculateCRC(), CFE_ES_CreateChildTask(), CFE_ES_DeleteChildTask(), CFE_ES_Exit↩App(), CFE_ES_GetAppInfo(), CFE_ES_RegisterCDS(), CFE_ES_ResetCFE(), and CFE_ES_RestartApp().

Here is the call graph for this function:



13.16 cfe/fsw/cfe-core/src/es/cfe_es_apps.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es.h"
#include "cfe_psp.h"
#include "cfe_es_global.h"
#include "cfe_es_apps.h"
#include "cfe_es_log.h"
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
```

Data Structures

- struct CFE_ES_CleanupState_t

Macros

- #define ES_START_BUFF_SIZE 128

Functions

- void CFE_ES_StartApplications (uint32 ResetType, const char *StartFilePath)
- int32 CFE_ES_ParseFileEntry (const char **TokenList, uint32 NumTokens)
- int32 CFE_ES_AppCreate (uint32 *ApplicationIdPtr, const char *FileName, const void *EntryPointData, const char *AppName, uint32 Priority, uint32 StackSize, uint32 ExceptionAction)
- int32 CFE_ES_LoadLibrary (uint32 *LibraryIdPtr, const char *FileName, const void *EntryPointData, const char *LibName)
- void CFE_ES_ScanAppTable (void)
- void CFE_ES_ProcessControlRequest (uint32 AppID)
- int32 CFE_ES_CleanUpApp (uint32 AppId)
- void CFE_ES_CleanupObjectCallback (uint32 ObjectId, void *arg)
- int32 CFE_ES_CleanupTaskResources (uint32 TaskId)
- void CFE_ES_CountObjectCallback (uint32 ObjectId, void *arg)
- int32 CFE_ES_ListResourcesDebug (void)
- void CFE_ES_GetAppInfoInternal (uint32 AppId, CFE_ES_AppInfo_t *AppInfoPtr)

13.16.1 Macro Definition Documentation

13.16.1.1 ES_START_BUFF_SIZE

```
#define ES_START_BUFF_SIZE 128
```

Definition at line 53 of file cfe_es_apps.c.

Referenced by CFE_ES_StartApplications().

13.16.2 Function Documentation

13.16.2.1 CFE_ES_AppCreate()

```
int32 CFE_ES_AppCreate (
            uint32 * ApplicationIdPtr,
            const char * FileName,
            const void * EntryPointData,
            const char * AppName,
            uint32 Priority,
            uint32 StackSize,
            uint32 ExceptionAction )
```

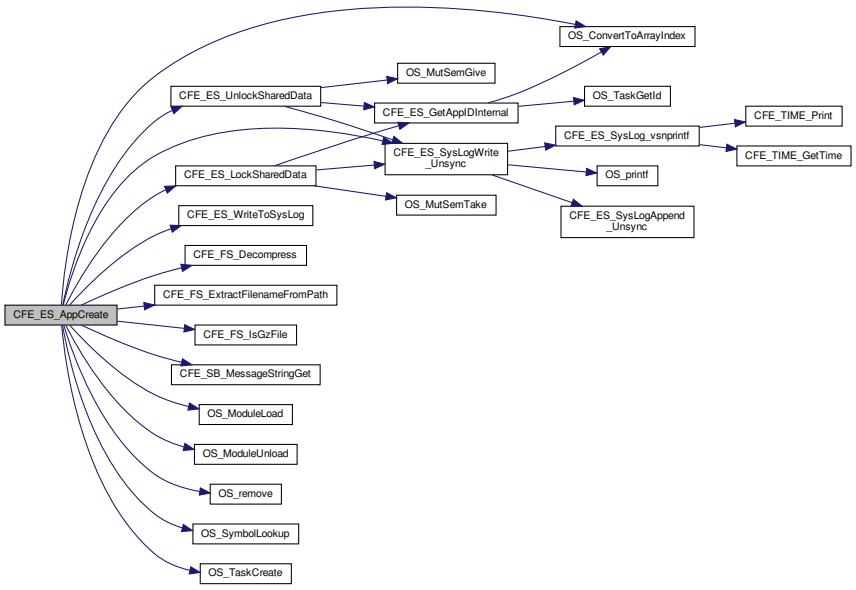
Definition at line 358 of file cfe_es_apps.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_TaskRecord_t::AppId, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_EARLY_INIT, CFE_ES_AppState_UNDEFINED, CFE_ES_AppType_EXTERNAL, CFE_ES_ERR_APP_CREATE, CFE_ES_Global,

CFE_ES_LockSharedData(), CFE_ES_RunStatus_APP_RUN, CFE_ES_SysLogWrite_Unsync(), CFE_ES_Unlock↩ SharedData(), CFE_ES_WriteToSysLog(), CFE_FS_Decompress(), CFE_FS_ExtractFilenameFromPath(), CFE_F↩ S_IsGzFile(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams↩ _t::EntryPoint, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_AppStartParams_t::FileName, CFE_ES_↩ MainTaskInfo_t::MainTaskId, CFE_ES_MainTaskInfo_t::MainTaskName, CFE_ES_AppStartParams_t::ModuleId, C↩ FE_ES_AppStartParams_t::Name, NULL, OS_ConvertToArrayIndex(), OS_FP_ENABLED, OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_ModuleLoad(), OS_ModuleUnload(), OS_remove(), OS_SUCCESS, OS_SymbolLookup(), OS_TaskCreate(), CFE_ES_AppStartParams_t::Priority, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t↩ ::RegisteredExternalApps, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppStartParams_t::StackSize, CFE_ES↩ _AppStartParams_t::StartAddress, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_E↩ S_AppRecord_t::TaskInfo, CFE_ES_TaskRecord_t::TaskName, CFE_ES_Global_t::TaskTable, and CFE_ES_App↩ Record_t::Type.

Referenced by CFE_ES_ParseFileEntry(), CFE_ES_ProcessControlRequest(), and CFE_ES_StartAppCmd().

Here is the call graph for this function:



13.16.2.2 CFE_ES_CleanUpApp()

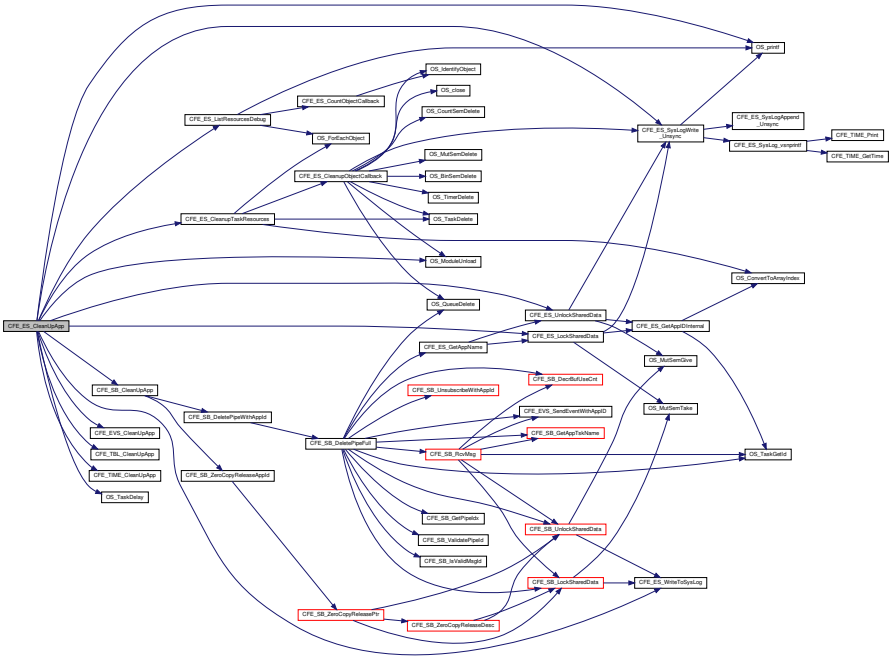
```
int32 CFE_ES_CleanUpApp (
            uint32 AppId )
```

Definition at line 1186 of file cfe_es_apps.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_APP_CLEANUP_ERR, CFE_ES_AppState_UNDEFINED, CFE_ES_AppType_EXTERNAL, CFE_ES_CleanupTaskResources(), CFE_ES_Global, CFE_ES_ListResourcesDebug(), CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_EVS_CleanUpApp(), CFE_SB_CleanUpApp(), CFE_SUCCESS, CFE_TBL_CleanUpApp(), CFE_TIME_CleanUpApp(), CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_AppStartParams_t::ModuleId, OS_ERROR, OS_MAX_TASKS, OS_ModuleUnload(), OS_printf(), OS_TaskDelay(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredExternalApps, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_Global_t::TaskTable, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ProcessControlRequest().

Here is the call graph for this function:



13.16.2.3 CFE_ES_CleanupObjectCallback()

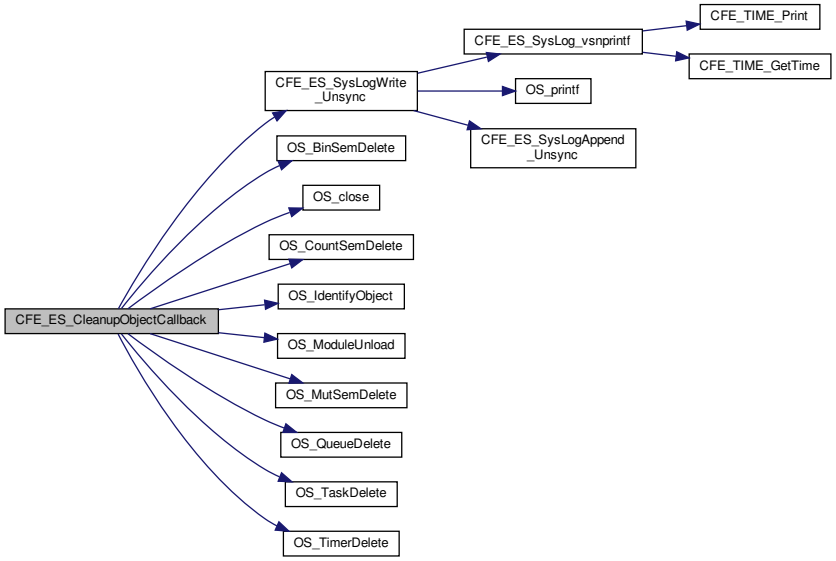
```
void CFE_ES_CleanupObjectCallback (
            uint32 ObjectId,
            void * arg )
```

Definition at line 1324 of file cfe_es_apps.c.

References CFE_ES_APP_CLEANUP_ERR, CFE_ES_BIN_SEM_DELETE_ERR, CFE_ES_COUNT_SEM_DELETE_ERR, CFE_ES_ERR_CHILD_TASK_DELETE, CFE_ES_MUT_SEM_DELETE_ERR, CFE_ES_QUEUE_DELETE_ERR, CFE_ES_SysLogWrite_Unsync(), CFE_ES_TIMER_DELETE_ERR, CFE_SUCCESS, CFE_ES_CleanupState_t::DeletedObjects, CFE_ES_CleanupState_t::FoundObjects, OS_BinSemDelete(), OS_close(), OS_CountSemDelete(), OS_ERROR, OS_IdentifyObject(), OS_ModuleUnload(), OS_MutSemDelete(), OS_OBJECT_TYPE_OS_BINSEM, OS_OBJECT_TYPE_OS_COUNTSEM, OS_OBJECT_TYPE_OS_MODULE, OS_OBJECT_TYPE_OS_MUTEX, OS_OBJECT_TYPE_OS_QUEUE, OS_OBJECT_TYPE_OS_STREAM, OS_OBJECT_TYPE_OS_TASK, OS_OBJECT_TYPE_OS_TIMECB, OS_QueueDelete(), OS_SUCCESS, OS_TaskDelete(), OS_TimerDelete(), and CFE_ES_CleanupState_t::OverallStatus.

Referenced by CFE_ES_CleanupTaskResources().

Here is the call graph for this function:



13.16.2.4 CFE_ES_CleanupTaskResources()

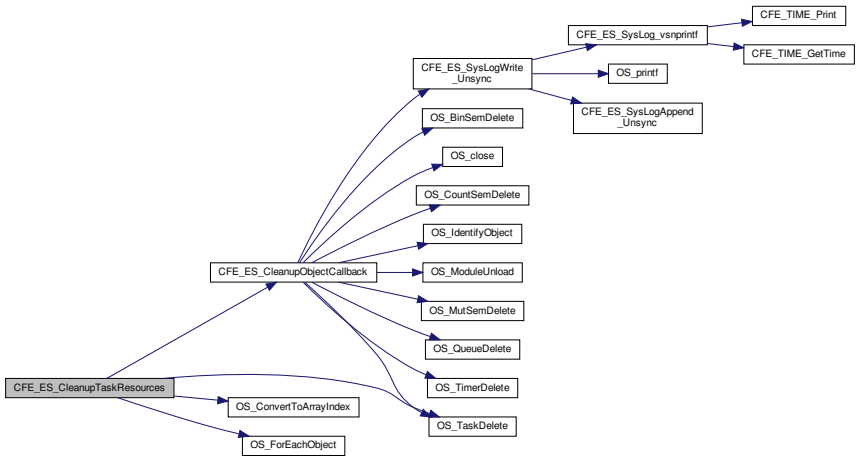
```
int32 CFE_ES_CleanupTaskResources (
            uint32 TaskId )
```

Definition at line 1422 of file cfe_es_apps.c.

References CFE_ES_APP_CLEANUP_ERR, CFE_ES_CleanupObjectCallback(), CFE_ES_Global, CFE_ES_TASK_DELETE_ERR, CFE_SUCCESS, CFE_ES_CleanupState_t::DeletedObjects, CFE_ES_CleanupState_t::ErrorFlag, CFE_ES_CleanupState_t::FoundObjects, OS_ConvertToArrayIndex(), OS_ForEachObject(), OS_SUCCESS, OS_TaskDelete(), CFE_ES_CleanupState_t::OverallStatus, CFE_ES_CleanupState_t::PrevFoundObjects, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_CleanUpApp().

Here is the call graph for this function:



13.16.2.5 CFE_ES_CountObjectCallback()

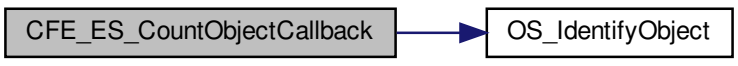
```
void CFE_ES_CountObjectCallback (
            uint32 ObjectId,
            void * arg )
```

Definition at line 1490 of file cfe_es_apps.c.

References OS_IdentifyObject(), and OS_OBJECT_TYPE_USER.

Referenced by CFE_ES_ListResourcesDebug().

Here is the call graph for this function:



13.16.2.6 CFE_ES_GetAppInfoInternal()

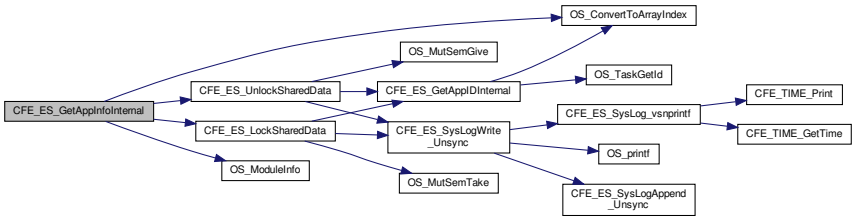
```
void CFE_ES_GetAppInfoInternal (
            uint32 AppId,
            CFE_ES_AppInfo_t * AppInfoPtr )
```

Definition at line 1537 of file cfe_es_apps.c.

References OS_module_prop_t::addr, CFE_ES_AppInfo_t::AddressesAreValid, CFE_ES_TaskRecord_t::AppId, CFE_ES_AppInfo_t::AppId, CFE_ES_Global_t::AppTable, OS_module_address_t::bss_address, OS_module_address_t::bss_size, CFE_ES_AppInfo_t::BSSAddress, CFE_ES_AppInfo_t::BSSSize, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_SB_SET_MEMADDR, OS_module_address_t::code_address, OS_module_address_t::code_size, CFE_ES_AppInfo_t::CodeAddress, CFE_ES_AppInfo_t::CodeSize, OS_module_address_t::data_address, OS_module_address_t::data_size, CFE_ES_AppInfo_t::DataAddress, CFE_ES_AppInfo_t::DataSize, CFE_ES_AppStartParams_t::EntryPoint, CFE_ES_AppInfo_t::EntryPoint, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_AppInfo_t::ExceptionAction, CFE_ES_TaskRecord_t::ExecutionCounter, CFE_ES_AppInfo_t::ExecutionCounter, CFE_ES_AppStartParams_t::FileName, CFE_ES_AppInfo_t::FileName, CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_AppInfo_t::MainTaskId, CFE_ES_MainTaskInfo_t::MainTaskName, CFE_ES_AppInfo_t::MainTaskName, CFE_ES_AppStartParams_t::ModuleId, CFE_ES_AppInfo_t::ModuleId, CFE_ES_AppStartParams_t::Name, CFE_ES_AppInfo_t::Name, CFE_ES_AppInfo_t::NumOfChildTasks, OS_ConvertToArrayIndex(), OS_MAX_TASKS, OS_ModuleInfo(), OS_SUCCESS, CFE_ES_AppStartParams_t::Priority, CFE_ES_AppInfo_t::Priority, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_AppStartParams_t::StackSize, CFE_ES_AppInfo_t::StackSize, CFE_ES_AppStartParams_t::StartAddress, CFE_ES_AppInfo_t::StartAddress, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_Global_t::TaskTable, CFE_ES_AppRecord_t::Type, CFE_ES_AppInfo_t::Type, and OS_module_address_t::valid.

Referenced by CFE_ES_GetAppInfo(), CFE_ES_QueryAllCmd(), and CFE_ES_QueryOneCmd().

Here is the call graph for this function:



13.16.2.7 CFE_ES_ListResourcesDebug()

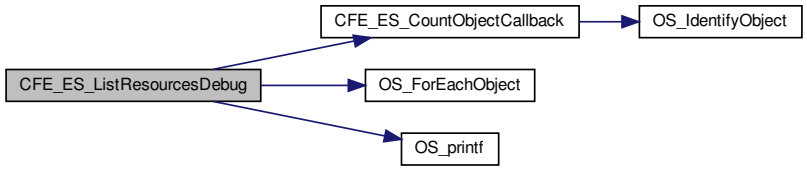
```
int32 CFE_ES_ListResourcesDebug (
            void  )
```

Definition at line 1511 of file cfe_es_apps.c.

References CFE_ES_CountObjectCallback(), CFE_SUCCESS, OS_ForEachObject(), OS_OBJECT_TYPE_OS_BINSEM, OS_OBJECT_TYPE_OS_COUNTSEM, OS_OBJECT_TYPE_OS_MUTEX, OS_OBJECT_TYPE_OS_QUEUE, OS_OBJECT_TYPE_OS_STREAM, OS_OBJECT_TYPE_OS_TASK, OS_OBJECT_TYPE_USER, and OS_printf().

Referenced by CFE_ES_CleanUpApp().

Here is the call graph for this function:



13.16.2.8 CFE_ES_LoadLibrary()

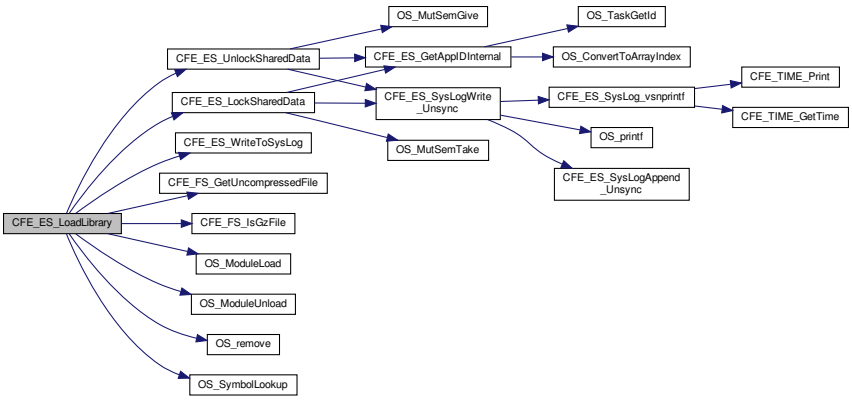
```
int32 CFE_ES_LoadLibrary (
            uint32 * LibraryIdPtr,
            const char * FileName,
            const void * EntryPointData,
            const char * LibName )
```

Definition at line 665 of file cfe_es_apps.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_ERR_LOAD_LIB, CFE_ES_Global, CFE_ES_LIB_ALREADY_↩ LOADED, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_FS_Get↩ UncompressedFile(), CFE_FS_IsGzFile(), CFE_PLATFORM_ES_MAX_LIBRARIES, CFE_PLATFORM_ES_RAM_↩ DISK_MOUNT_STRING, CFE_SUCCESS, CFE_ES_LibRecord_t::LibName, CFE_ES_Global_t::LibTable, NULL, O↩ S_MAX_PATH_LEN, OS_ModuleLoad(), OS_ModuleUnload(), OS_remove(), OS_SUCCESS, OS_SymbolLookup(), CFE_ES_LibRecord_t::RecordUsed, and CFE_ES_Global_t::RegisteredLibs.

Referenced by CFE_ES_ParseFileEntry().

Here is the call graph for this function:



13.16.2.9 CFE_ES_ParseFileEntry()

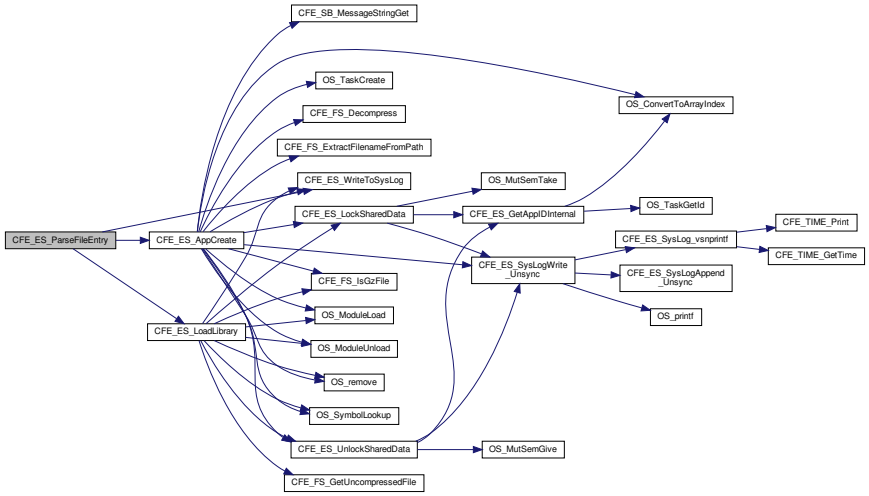
```
int32 CFE_ES_ParseFileEntry (
            const char ** TokenList,
            uint32 NumTokens )
```

Definition at line 265 of file cfe_es_apps.c.

References CFE_ES_AppCreate(), CFE_ES_ERR_APP_CREATE, CFE_ES_ExceptionAction_PROC_RESTART, CFE_ES_ExceptionAction_RESTART_APP, CFE_ES_LoadLibrary(), CFE_ES_WriteToSysLog(), and NULL.

Referenced by CFE_ES_StartApplications().

Here is the call graph for this function:



13.16.2.10 CFE_ES_ProcessControlRequest()

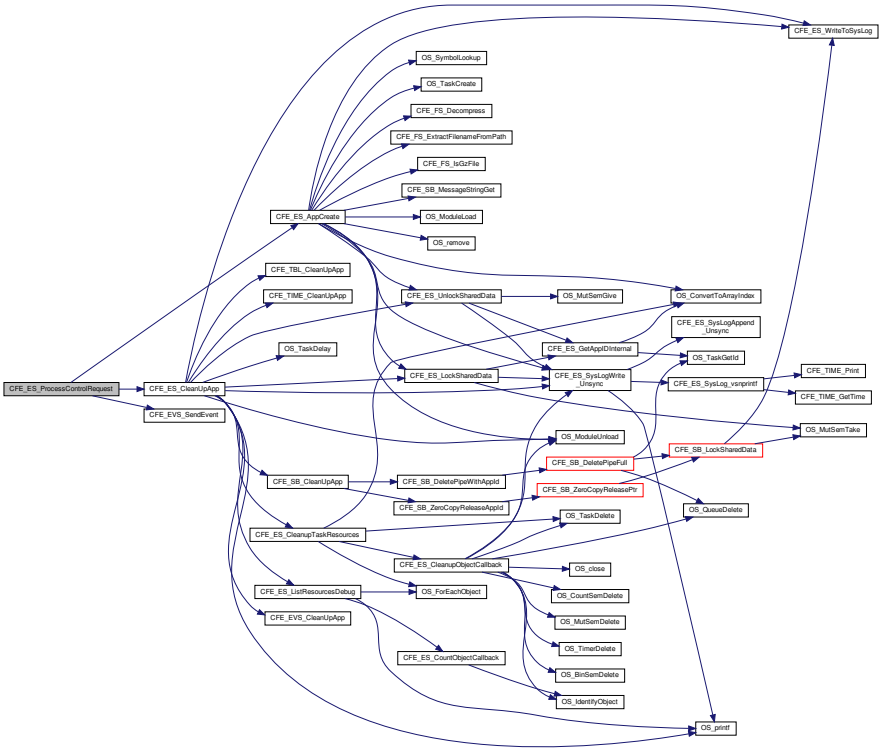
```
void CFE_ES_ProcessControlRequest (
            uint32 AppID )
```

Definition at line 1002 of file cfe_es_apps.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_Global_t::AppTable, CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_ERREXIT_APP_ERR_EID, CFE_ES_ERREXIT_APP_INF_EID, CFE_ES_EXIT_APP_ERR_EID, CFE_ES_EXIT_APP_INF_EID, CFE_ES_Global, CFE_ES_PCR_ERR1_EID, CFE_ES_PCR_ERR2_EID, CFE_ES_RELOAD_APP_ERR3_EID, CFE_ES_RELOAD_APP_ERR4_EID, CFE_ES_RELOAD_APP_INF_EID, CFE_ES_RESTART_APP_ERR3_EID, CFE_ES_RESTART_APP_ERR4_EID, CFE_ES_RESTART_APP_INF_EID, CFE_ES_RunStatus_APP_ERROR, CFE_ES_RunStatus_APP_EXIT, CFE_ES_RunStatus_SYS_DELETE, CFE_ES_RunStatus_SYS_EXCEPTION, CFE_ES_RunStatus_SYS_RELOAD, CFE_ES_RunStatus_SYS_RESTART, CFE_ES_STOP_ERR3_EID, CFE_ES_STOP_INF_EID, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::EntryPoint, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_AppStartParams_t::FileName, CFE_ES_AppStartParams_t::Name, CFE_ES_AppStartParams_t::Priority, CFE_ES_AppStartParams_t::StackSize, and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_ScanAppTable().

Here is the call graph for this function:



13.16.2.11 CFE_ES_ScanAppTable()

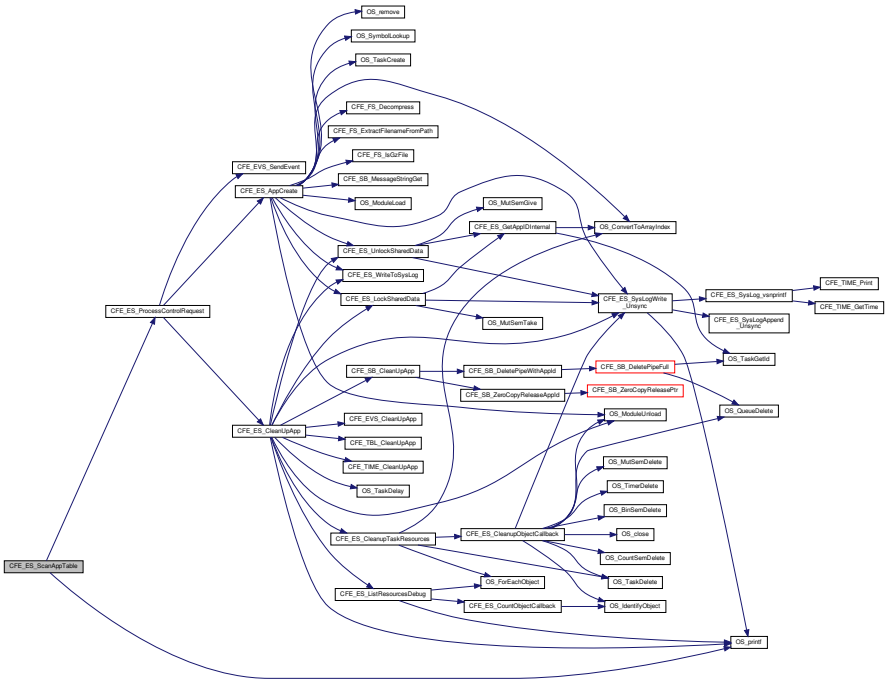
```
void CFE_ES_ScanAppTable (
            void  )
```

Definition at line 938 of file cfe_es_apps.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CF↩E_ES_AppState_STOPPED, CFE_ES_AppState_WAITING, CFE_ES_AppType_EXTERNAL, CFE_ES_Global, CF↩E_ES_ProcessControlRequest(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_ES_AppRecord_t::ControlReq, OS_printf(), and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_TaskMain().

Here is the call graph for this function:



13.16.2.12 CFE_ES_StartApplications()

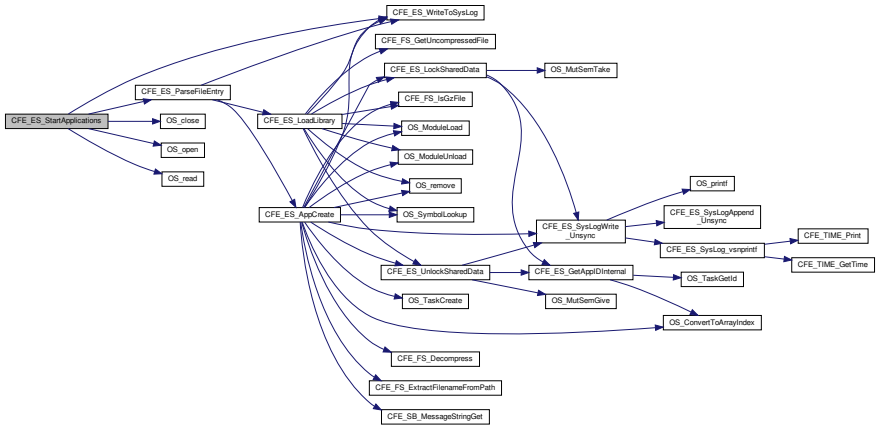
```
void CFE_ES_StartApplications (
            uint32 ResetType,
            const char * StartFilePath )
```

Definition at line 78 of file cfe_es_apps.c.

References CFE_ES_ParseFileEntry(), CFE_ES_STARTSCRIPT_MAX_TOKENS_PER_LINE, CFE_ES_WriteTo↩SysLog(), CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE, CFE_PSP_RST_TYPE_PROCESSOR, ES_START↩_BUFF_SIZE, OS_close(), OS_FS_ERROR, OS_open(), and OS_read().

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.17 cfe/fsw/cfe-core/src/es/cfe_es_apps.h File Reference

```
#include "common_types.h"
#include "osapi.h"
```

Data Structures

- struct CFE_ES_ControlReq_t
- struct CFE_ES_AppStartParams_t
- struct CFE_ES_MainTaskInfo_t
- struct CFE_ES_AppRecord_t
- struct CFE_ES_TaskRecord_t
- struct CFE_ES_LibRecord_t

Macros

- #define CFE_ES_STARTSCRIPT_MAX_TOKENS_PER_LINE 8

Functions

- void CFE_ES_StartApplications (uint32 ResetType, const char ∗StartFilePath)
- int32 CFE_ES_ParseFileEntry (const char ∗∗TokenList, uint32 NumTokens)
- void CFE_ES_SetAppState (uint32 AppID, uint32 TargetState)
- int32 CFE_ES_AppCreate (uint32 ∗ApplicationIdPtr, const char ∗FileName, const void ∗EntryPointData, const char ∗AppName, uint32 Priority, uint32 StackSize, uint32 ExceptionAction)
- int32 CFE_ES_LoadLibrary (uint32 ∗LibraryIdPtr, const char ∗FileName, const void ∗EntryPointData, const char ∗LibName)
- int32 CFE_ES_AppGetList (uint32 AppIdArray[ ], uint32 ArraySize)

- int32 CFE_ES_AppDumpAllInfo (void)
- void CFE_ES_ScanAppTable (void)
- void CFE_ES_ProcessControlRequest (uint32 AppID)
- int32 CFE_ES_CleanUpApp (uint32 AppId)
- int32 CFE_ES_CleanupTaskResources (uint32 TaskId)
- int32 CFE_ES_ListResourcesDebug (void)
- void CFE_ES_GetAppInfoInternal (uint32 AppId, CFE_ES_AppInfo_t *AppInfoPtr)

13.17.1 Macro Definition Documentation

13.17.1.1 CFE_ES_STARTSCRIPT_MAX_TOKENS_PER_LINE

```
#define CFE_ES_STARTSCRIPT_MAX_TOKENS_PER_LINE 8
```

Definition at line 48 of file cfe_es_apps.h.

Referenced by CFE_ES_StartApplications().

13.17.2 Function Documentation

13.17.2.1 CFE_ES_AppCreate()

```
int32 CFE_ES_AppCreate (
            uint32 * ApplicationIdPtr,
            const char * FileName,
            const void * EntryPointData,
            const char * AppName,
            uint32 Priority,
            uint32 StackSize,
            uint32 ExceptionAction )
```

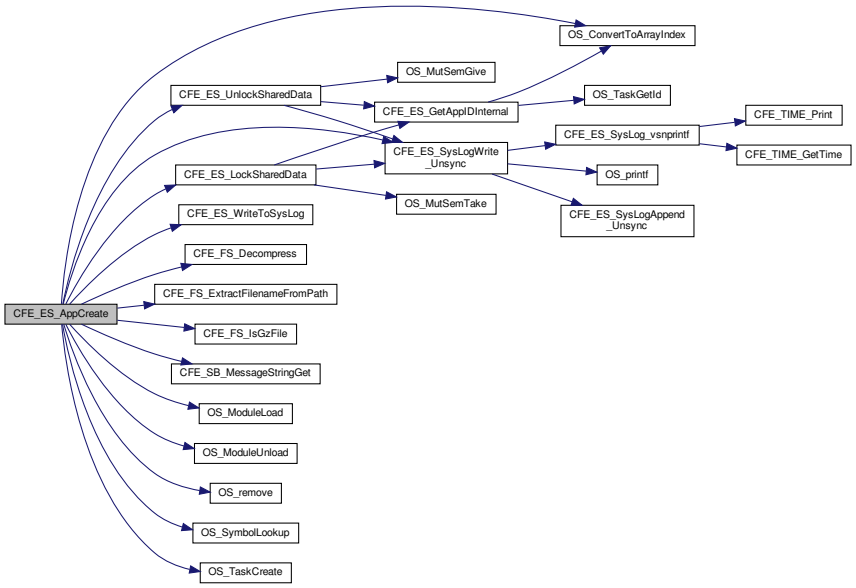
Definition at line 358 of file cfe_es_apps.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_TaskRecord_t::AppId, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_EARLY_INIT, CFE_ES_AppState_UNDEFINED, CFE_ES_AppType_EXTERNAL, CFE_ES_ERR_APP_CREATE, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_APP_RUN, CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_FS_Decompress(), CFE_FS_ExtractFilenameFromPath(), CFE_FS_IsGzFile(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::EntryPoint, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_AppStartParams_t::FileName, CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_MainTaskInfo_t::MainTaskName, CFE_ES_AppStartParams_t::ModuleId, CFE_ES_AppStartParams_t::Name, NULL, OS_ConvertToArrayIndex(), OS_FP_ENABLED, OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_ModuleLoad(), OS_ModuleUnload(), OS_remove(), OS_SUCCESS, OS_SymbolLookup(),

OS_TaskCreate(), CFE_ES_AppStartParams_t::Priority, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredExternalApps, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppStartParams_t::StackSize, CFE_ES_AppStartParams_t::StartAddress, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_TaskRecord_t::TaskName, CFE_ES_Global_t::TaskTable, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ParseFileEntry(), CFE_ES_ProcessControlRequest(), and CFE_ES_StartAppCmd().

Here is the call graph for this function:



13.17.2.2 CFE_ES_AppDumpAllInfo()

```
int32 CFE_ES_AppDumpAllInfo (
            void  )
```

13.17.2.3 CFE_ES_AppGetList()

```
int32 CFE_ES_AppGetList (
            uint32 AppIdArray[],
            uint32 ArraySize )
```

13.17.2.4 CFE_ES_CleanUpApp()

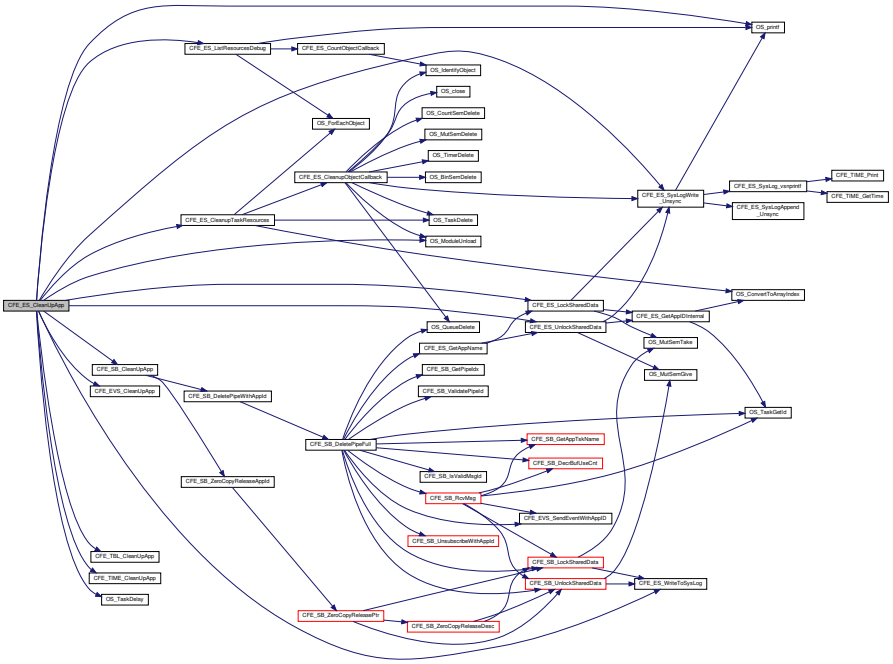
```
int32 CFE_ES_CleanUpApp (
            uint32 AppId )
```

Definition at line 1186 of file cfe_es_apps.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_APP_CLEANUP_ERR, CFE_ES_AppState_UNDEFINED, CFE_ES_AppType_EXTERNAL, CFE_ES_CleanupTaskResources(), CFE_ES_Global, CFE_ES_ListResourcesDebug(), CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_EVS_CleanUpApp(), CFE_SB_CleanUpApp(), CFE_SUCCESS, CFE_TBL_CleanUpApp(), CFE_TIME_CleanUpApp(), CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_AppStartParams_t::ModuleId, OS_ERROR, OS_MAX_TASKS, OS_ModuleUnload(), OS_printf(), OS_TaskDelay(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredExternalApps, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_Global_t::TaskTable, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ProcessControlRequest().

Here is the call graph for this function:



13.17.2.5 CFE_ES_CleanupTaskResources()

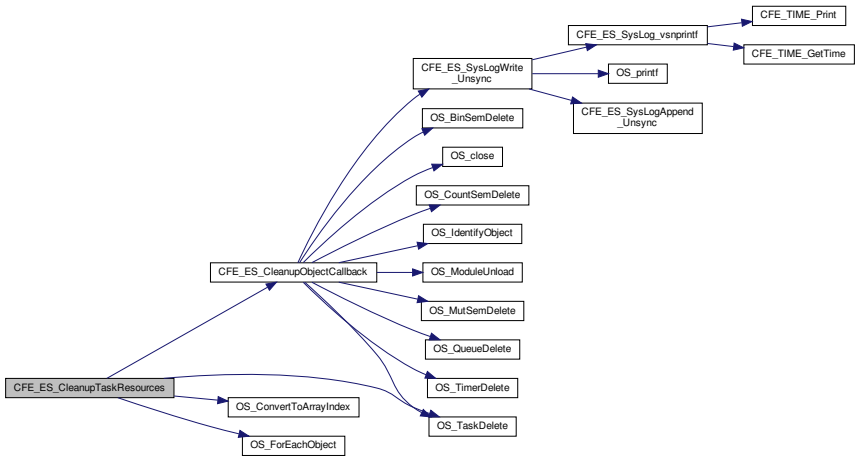
```
int32 CFE_ES_CleanupTaskResources (
            uint32 TaskId )
```

Definition at line 1422 of file cfe_es_apps.c.

References CFE_ES_APP_CLEANUP_ERR, CFE_ES_CleanupObjectCallback(), CFE_ES_Global, CFE_ES_TASK_DELETE_ERR, CFE_SUCCESS, CFE_ES_CleanupState_t::DeletedObjects, CFE_ES_CleanupState_t::ErrorFlag, CFE_ES_CleanupState_t::FoundObjects, OS_ConvertToArrayIndex(), OS_ForEachObject(), OS_SUCCESS, OS_TaskDelete(), CFE_ES_CleanupState_t::OverallStatus, CFE_ES_CleanupState_t::PrevFoundObjects, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_CleanUpApp().

Here is the call graph for this function:



13.17.2.6 CFE_ES_GetAppInfoInternal()

```
void CFE_ES_GetAppInfoInternal (
            uint32 AppId,
            CFE_ES_AppInfo_t * AppInfoPtr )
```

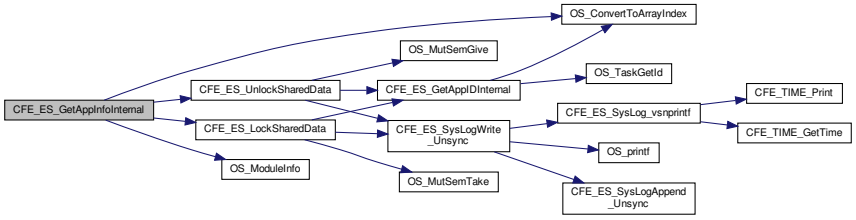
Definition at line 1537 of file cfe_es_apps.c.

References OS_module_prop_t::addr, CFE_ES_AppInfo_t::AddressesAreValid, CFE_ES_TaskRecord_t::AppId, CFE_ES_AppInfo_t::AppId, CFE_ES_Global_t::AppTable, OS_module_address_t::bss_address, OS_module_address_t::bss_size, CFE_ES_AppInfo_t::BSSAddress, CFE_ES_AppInfo_t::BSSSize, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_SB_SET_MEMADDR, OS_module_address_t::code_address, OS_module_address_t::code_size, CFE_ES_AppInfo_t::CodeAddress, CFE_ES_AppInfo_t::CodeSize, OS_module_address_t::data_address, OS_module_address_t::data_size, CFE_ES_AppInfo_t::DataAddress, CFE_ES_AppInfo_t::DataSize, CFE_ES_AppStartParams_t::EntryPoint, CFE_ES_AppInfo_t::EntryPoint, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_AppInfo_t::ExceptionAction, CFE_ES_TaskRecord_t::ExecutionCounter, CFE_ES_AppInfo_t::ExecutionCounter, CFE_ES_AppStartParams_t::FileName, CFE_ES_AppInfo_t::FileName, CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_AppInfo_t::MainTaskId, CFE_ES_MainTaskInfo_t::MainTaskName, CFE_ES_AppInfo_t::MainTaskName, CFE_ES_AppStartParams_t::ModuleId, CFE_ES_AppInfo_t::ModuleId, CFE_ES_AppStartParams_t::Name, CFE_ES_AppInfo_t::Name, CFE_ES_AppInfo_t::NumOfChildTasks, OS_ConvertToArrayIndex(), OS_MAX_TASKS, OS_ModuleInfo(), OS_SUCCESS, CFE_ES_AppStartParams_t::Priority, CFE_ES_AppInfo_t::Priority,

CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_AppStartParams_t::StackSize, CFE_ES_AppInfo_t::StackSize, CF↩E_ES_AppStartParams_t::StartAddress, CFE_ES_AppInfo_t::StartAddress, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_Global_t::TaskTable, CFE_ES_App↩Record_t::Type, CFE_ES_AppInfo_t::Type, and OS_module_address_t::valid.

Referenced by CFE_ES_GetAppInfo(), CFE_ES_QueryAllCmd(), and CFE_ES_QueryOneCmd().

Here is the call graph for this function:



13.17.2.7 CFE_ES_ListResourcesDebug()

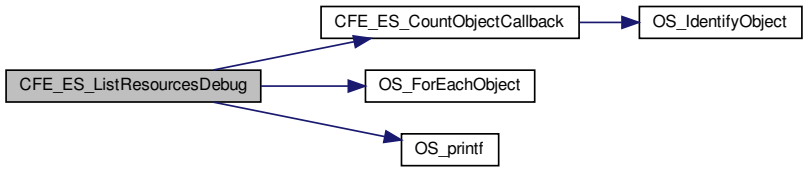
```
int32 CFE_ES_ListResourcesDebug (
            void  )
```

Definition at line 1511 of file cfe_es_apps.c.

References CFE_ES_CountObjectCallback(), CFE_SUCCESS, OS_ForEachObject(), OS_OBJECT_TYPE_OS_BI↩NSEM, OS_OBJECT_TYPE_OS_COUNTSEM, OS_OBJECT_TYPE_OS_MUTEX, OS_OBJECT_TYPE_OS_QUEUE, OS_OBJECT_TYPE_OS_STREAM, OS_OBJECT_TYPE_OS_TASK, OS_OBJECT_TYPE_USER, and OS_printf().

Referenced by CFE_ES_CleanUpApp().

Here is the call graph for this function:



13.17.2.8 CFE_ES_LoadLibrary()

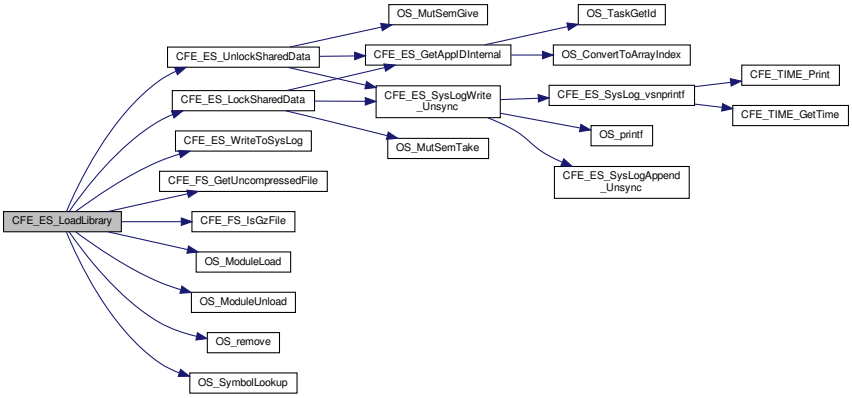
```
int32 CFE_ES_LoadLibrary (
            uint32 * LibraryIdPtr,
            const char * FileName,
            const void * EntryPointData,
            const char * LibName )
```

Definition at line 665 of file cfe_es_apps.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_ERR_LOAD_LIB, CFE_ES_Global, CFE_ES_LIB_ALREADY_↩ LOADED, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_FS_Get↩ UncompressedFile(), CFE_FS_IsGzFile(), CFE_PLATFORM_ES_MAX_LIBRARIES, CFE_PLATFORM_ES_RAM_↩ DISK_MOUNT_STRING, CFE_SUCCESS, CFE_ES_LibRecord_t::LibName, CFE_ES_Global_t::LibTable, NULL, O↩ S_MAX_PATH_LEN, OS_ModuleLoad(), OS_ModuleUnload(), OS_remove(), OS_SUCCESS, OS_SymbolLookup(), CFE_ES_LibRecord_t::RecordUsed, and CFE_ES_Global_t::RegisteredLibs.

Referenced by CFE_ES_ParseFileEntry().

Here is the call graph for this function:



13.17.2.9 CFE_ES_ParseFileEntry()

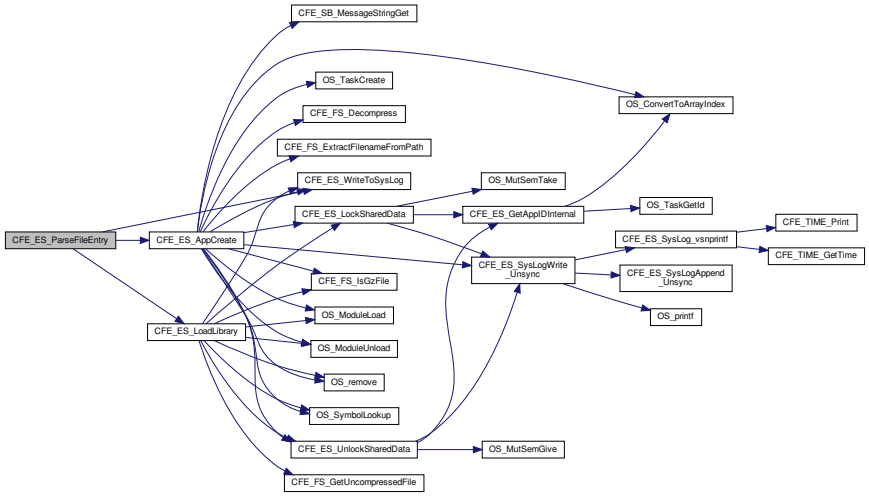
```
int32 CFE_ES_ParseFileEntry (
            const char ** TokenList,
            uint32 NumTokens )
```

Definition at line 265 of file cfe_es_apps.c.

References CFE_ES_AppCreate(), CFE_ES_ERR_APP_CREATE, CFE_ES_ExceptionAction_PROC_RESTART, C↩ FE_ES_ExceptionAction_RESTART_APP, CFE_ES_LoadLibrary(), CFE_ES_WriteToSysLog(), and NULL.

Referenced by CFE_ES_StartApplications().

Here is the call graph for this function:



13.17.2.10 CFE_ES_ProcessControlRequest()

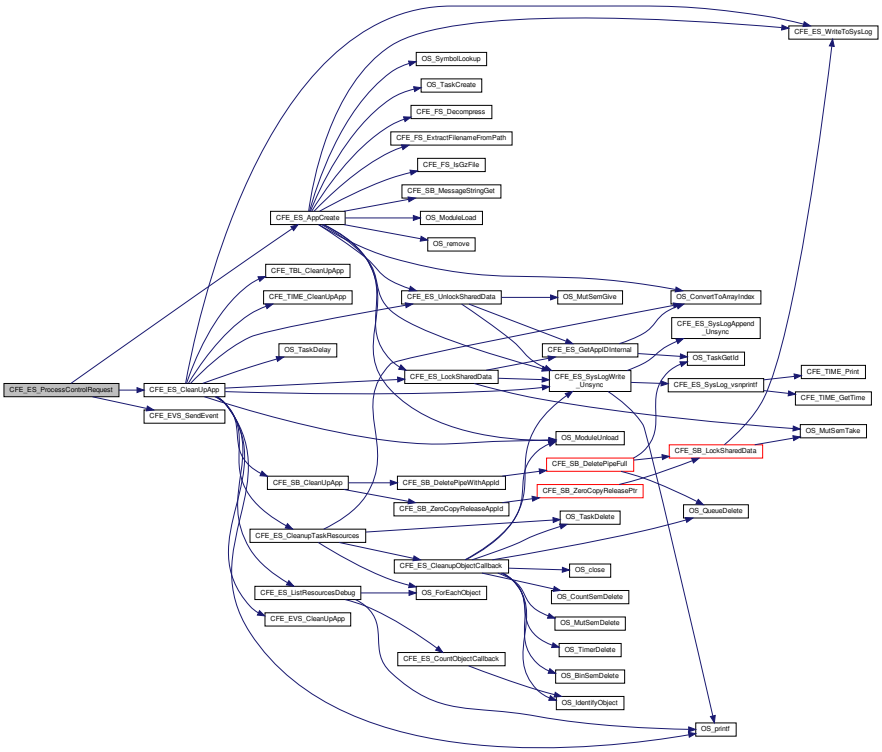
```
void CFE_ES_ProcessControlRequest (
            uint32 AppID )
```

Definition at line 1002 of file cfe_es_apps.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_Global_t::AppTable, CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_ERREXIT_APP_ERR_EID, CFE_ES_ERREXIT_APP_INF_EID, CFE_ES_EXIT_APP_ERR_EID, CFE_ES_EXIT_APP_INF_EID, CFE_ES_Global, CFE_ES_PCR_ERR1_EID, CFE_ES_PCR_ERR2_EID, CFE_ES_RELOAD_APP_ERR3_EID, CFE_ES_RELOAD_APP_ERR4_EID, CFE_ES_RELOAD_APP_INF_EID, CFE_ES_RESTART_APP_ERR3_EID, CFE_ES_RESTART_APP_ERR4_EID, CFE_ES_RESTART_APP_INF_EID, CFE_ES_RunStatus_APP_ERROR, CFE_ES_RunStatus_APP_EXIT, CFE_ES_RunStatus_SYS_DELETE, CFE_ES_RunStatus_SYS_EXCEPTION, CFE_ES_RunStatus_SYS_RELOAD, CFE_ES_RunStatus_SYS_RESTART, CFE_ES_STOP_ERR3_EID, CFE_ES_STOP_INF_EID, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::EntryPoint, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_AppStartParams_t::FileName, CFE_ES_AppStartParams_t::Name, CFE_ES_AppStartParams_t::Priority, CFE_ES_AppStartParams_t::StackSize, and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_ScanAppTable().

Here is the call graph for this function:



13.17.2.11 CFE_ES_ScanAppTable()

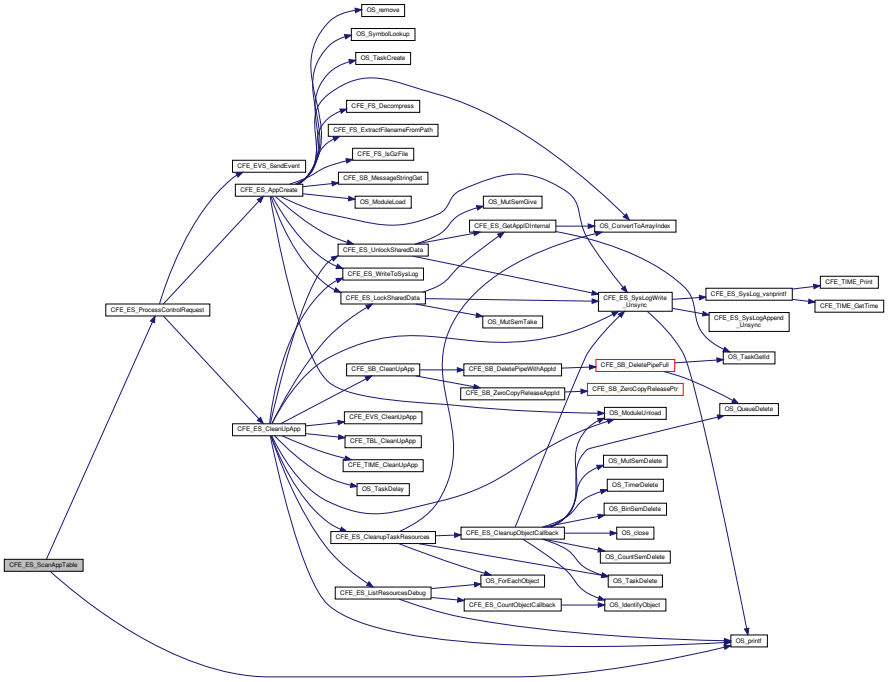
```
void CFE_ES_ScanAppTable (
            void  )
```

Definition at line 938 of file cfe_es_apps.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CF↩E_ES_AppState_STOPPED, CFE_ES_AppState_WAITING, CFE_ES_AppType_EXTERNAL, CFE_ES_Global, CF↩E_ES_ProcessControlRequest(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_ES_AppRecord_t::ControlReq, OS_printf(), and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_TaskMain().

Here is the call graph for this function:



13.17.2.12 CFE_ES_SetAppState()

```
void CFE_ES_SetAppState (
            uint32 AppID,
            uint32 TargetState )
```

Definition at line 191 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_MAX, CFE_ES_↩ AppState_UNDEFINED, and CFE_ES_Global.

Referenced by CFE_ES_DeleteApp(), CFE_ES_ExitApp(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES↩ _RunLoop(), and CFE_ES_WaitForSystemState().

13.17.2.13 CFE_ES_StartApplications()

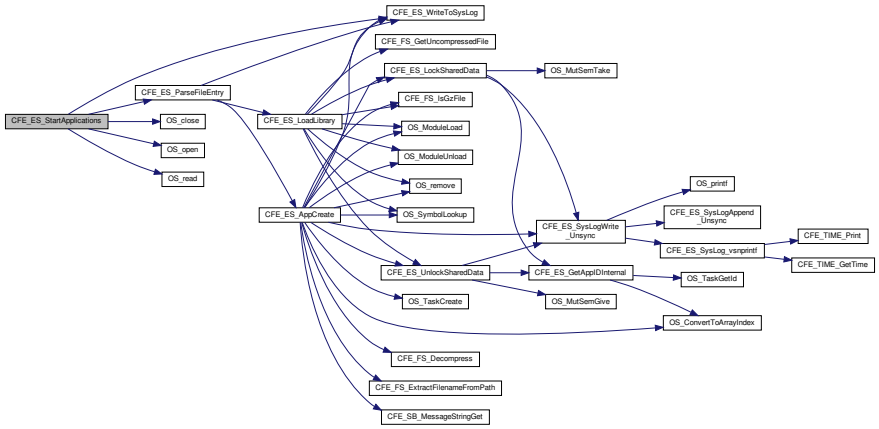
```
void CFE_ES_StartApplications (
            uint32 ResetType,
            const char * StartFilePath )
```

Definition at line 78 of file cfe_es_apps.c.

References CFE_ES_ParseFileEntry(), CFE_ES_STARTSCRIPT_MAX_TOKENS_PER_LINE, CFE_ES_WriteTo↩ SysLog(), CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE, CFE_PSP_RST_TYPE_PROCESSOR, ES_START↩ _BUFF_SIZE, OS_close(), OS_FS_ERROR, OS_open(), and OS_read().

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.18 cfe/fsw/cfe-core/src/es/cfe_es_cds.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es_apps.h"
#include "cfe_es_cds.h"
#include "cfe_es_global.h"
#include "cfe_es_log.h"
#include "cfe_psp.h"
#include "cfe_es_cds_mempool.h"
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
```

Macros

- #define CDS_REG_SIZE_OFFSET ((sizeof(CFE_ES_Global.CDSVars.ValidityField)+3) & 0xfffffffc)
- #define CDS_REG_OFFSET (((CDS_REG_SIZE_OFFSET + sizeof(CFE_ES_Global.CDSVars.MaxNumReg↩ Entries)) + 3) & 0xfffffffc)
- #define CDS_POOL_OFFSET (((CDS_REG_OFFSET + (CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES ∗ sizeof(CFE_ES_CDS_RegRec_t))) + 3) & 0xfffffffc)

Functions

- int32 CFE_ES_ValidateCDS (void)

  Determines whether a CDS currently exists.
- int32 CFE_ES_InitializeCDS (uint32 CDSSize)

  Initializes the contents of the CDS.
- int32 CFE_ES_InitCDSRegistry (void)

  Initializes the CDS Registry.
- int32 CFE_ES_RebuildCDS (void)

  Rebuilds memory pool for CDS and recovers existing registry.
- int32 CFE_ES_CDS_EarlyInit (void)

  Initializes CDS data constructs.
- int32 CFE_ES_RegisterCDSEx (CFE_ES_CDSHandle_t *HandlePtr, int32 BlockSize, const char *Name, bool CriticalTbl)

  Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
- int32 CFE_ES_UpdateCDSRegistry (void)

  Copies the local version of the CDS Registry to the actual CDS.
- int32 CFE_ES_CDS_ValidateAppID (uint32 *AppIdPtr)

  Validates the Application ID associated with calling Application.
- void CFE_ES_FormCDSName (char *FullCDSName, const char *CDSName, uint32 ThisAppId)

  Creates a Full CDS name from application name and CDS name.
- int32 CFE_ES_LockCDSRegistry (void)

  Locks access to the CDS Registry.
- int32 CFE_ES_UnlockCDSRegistry (void)

  Unlocks access to the CDS Registry.
- int32 CFE_ES_FindCDSInRegistry (const char *CDSName)

  Returns the Registry Index for the specified CDS Name.
- int32 CFE_ES_FindFreeCDSRegistryEntry (void)

  Locates a free slot in the CDS Registry.
- int32 CFE_ES_DeleteCDS (const char *CDSName, bool CalledByTblServices)

  Deletes the specified CDS from the CDS Registry and frees CDS Memory.

13.18.1 Macro Definition Documentation

13.18.1.1 CDS_POOL_OFFSET

```
#define CDS_POOL_OFFSET (((CDS_REG_OFFSET + (CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES * sizeof(CFE_ES↩
_CDS_RegRec_t))) + 3) & 0xfffffffc)
```

Definition at line 57 of file cfe_es_cds.c.

Referenced by CFE_ES_InitializeCDS().

13.18.1.2 CDS_REG_OFFSET

```
#define CDS_REG_OFFSET (((CDS_REG_SIZE_OFFSET + sizeof(CFE_ES_Global.CDSVars.MaxNumRegEntries)) +
3) & 0xfffffffc)
```

Definition at line 56 of file cfe_es_cds.c.

Referenced by CFE_ES_RebuildCDS(), and CFE_ES_UpdateCDSRegistry().

13.18.1.3 CDS_REG_SIZE_OFFSET

```
#define CDS_REG_SIZE_OFFSET ((sizeof(CFE_ES_Global.CDSVars.ValidityField)+3) & 0xfffffffc)
```

Definition at line 55 of file cfe_es_cds.c.

Referenced by CFE_ES_InitCDSRegistry(), and CFE_ES_RebuildCDS().

13.18.2 Function Documentation

13.18.2.1 CFE_ES_CDS_EarlyInit()

```
int32 CFE_ES_CDS_EarlyInit (
            void  )
```

Initializes the cFE core module API Library.

Description

Locates and validates any pre-existing CDS memory or initializes the memory as a fresh CDS.

Assumptions, External Events, and Notes:

None

SysLog Messages

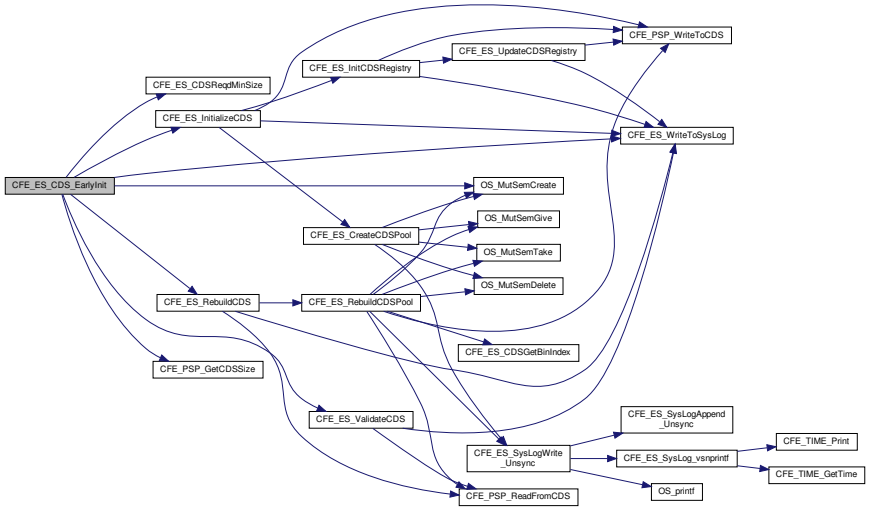
Returns

None

Definition at line 149 of file cfe_es_cds.c.

References CFE_ES_CDSVariables_t::CDSSize, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_INVALID, CFE_ES_CDS_MUT_REG_NAME, CFE_ES_CDS_MUT_REG_VALUE, CFE_ES_CDSReqdMinSize(), CFE_ES_Global, CFE_ES_InitializeCDS(), CFE_ES_RebuildCDS(), CFE_ES_ValidateCDS(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PSP_GetCDSSize(), CFE_PSP_SUCCESS, CFE_SUCCESS, CFE_ES_CDSVariables_t::MemPoolSize, OS_MutSemCreate(), CFE_ES_CDSVariables_t::RegistryMutex, and CFE_ES_CDSVariables_t::ValidityField.

Here is the call graph for this function:



13.18.2.2 CFE_ES_CDS_ValidateAppID()

```
int32 CFE_ES_CDS_ValidateAppID (
            uint32 * AppIdPtr )
```

Description

Validates Application ID of calling App. Validation consists of ensuring the AppID is between zero and CFE_PLATFORM_ES_MAX_APPLICATIONS.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppIdPtr*	Pointer to value that will hold AppID on return.
out	**AppIdPtr*	The AppID as obtained from CFE_ES_GetAppID

Return values

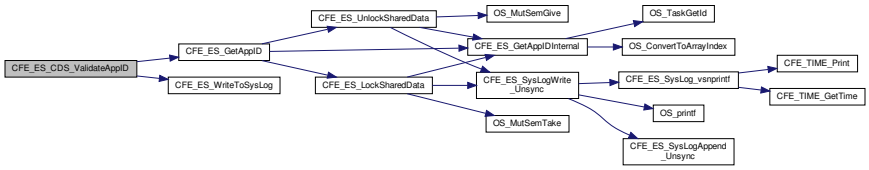
CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.

Definition at line 537 of file cfe_es_cds.c.

References CFE_ES_ERR_APPID, CFE_ES_GetAppID(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX↩ _APPLICATIONS, and CFE_SUCCESS.

Referenced by CFE_ES_RegisterCDS().

Here is the call graph for this function:



13.18.2.3 CFE_ES_DeleteCDS()

```
int32 CFE_ES_DeleteCDS (
            const char * CDSName,
            bool CalledByTblServices )
```

Description

Removes the record of the specified CDS from the CDS Registry and frees the associated CDS memory for future use.

Assumptions, External Events, and Notes:

None

Parameters

in	*CDSName*	- Pointer to character string containing complete CDS Name (of the format "AppName.CDSName").
in	*CalledByTblServices*	- Flag that identifies whether the CDS is supposed to be a Critical Table Image or not.

Returns

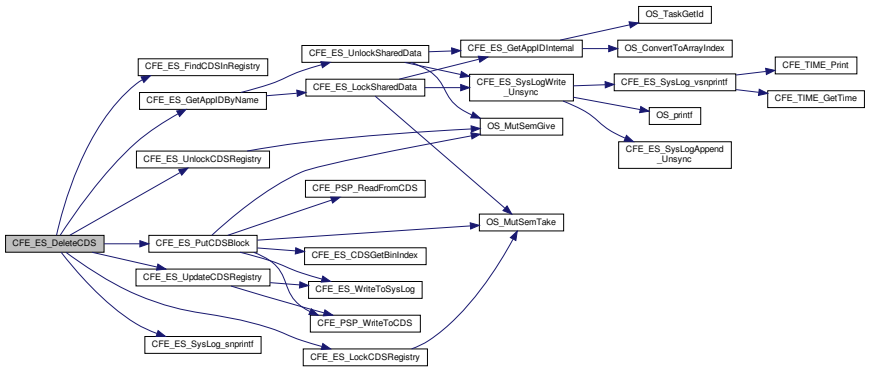
CFE_SUCCESS Operation was performed successfully
CFE_ES_CDS_WRONG_TYPE_ERR Occurs when Table Services is trying to delete a Critical Data Store that is not a Critical Table Image or when Executive Services is trying to delete a Critical Table Image.
CFE_ES_CDS_OWNER_ACTIVE_ERR Occurs when an attempt was made to delete a CDS when an application with the same name associated with the CDS is still present. CDSs can ONLY be deleted when Applications that created them are not present in the system.
CFE_ES_CDS_NOT_FOUND_ERR Occurs when a search of the Critical Data Store Registry does not find a critical data store with the specified name.
Any of the return values from CFE_ES_UpdateCDSRegistry
Any of the return values from CFE_ES_PutCDSBlock

Definition at line 752 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_NOT_FOUND, CFE_ES_CDS_NOT_FOUND_ERR, CFE_ES_CDS_OWNER_ACTIVE_ERR, CFE_ES_CDS_WRONG_TYPE_ERR, CFE_ES_ERR_APPNAME, CFE_ES_FindCDSInRegistry(), CFE_ES_GetAppIDByName(), CFE_ES_Global, CFE_ES_LockCDSRegistry(), CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_PutCDSBlock(), CFE_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UpdateCDSRegistry(), CFE_SUCCESS, CFE_ES_CDS_RegRec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, NULL, OS_MAX_API_NAME, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_DeleteCDSCmd().

Here is the call graph for this function:



13.18.2.4 CFE_ES_FindCDSInRegistry()

```
int32 CFE_ES_FindCDSInRegistry (
            const char * CDSName )
```

Description

Locates given CDS Name in the CDS Registry and returns the appropriate Registry Index.

Assumptions, External Events, and Notes:

None

Parameters

in	*CDSName*	- Pointer to character string containing complete CDS Name (of the format "AppName.CDSName").

Return values

CFE_ES_CDS_NOT_FOUND	or the Index into Registry for Table with specified name

Definition at line 636 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_NOT_FOUND, CFE_ES_Global, CFE_ES_CDSVariables_t::MaxNumRegEntries, CFE_ES_CDS_RegRec_t::Name, CFE_ES_CDSVariables_t::Registry, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

13.18.2.5 CFE_ES_FindFreeCDSRegistryEntry()

```
int32 CFE_ES_FindFreeCDSRegistryEntry (
            void  )
```

Description

Locates a free slot in the CDS Registry.

Assumptions, External Events, and Notes:

Note: This function assumes the registry has been locked.

Return values

CFE_ES_CDS_NOT_FOUND	or Index into CDS Registry of unused entry

Definition at line 669 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_NOT_FOUND, CFE_ES_Global, CFE_ES_CDSVariables_t::MaxNumRegEntries, CFE_ES_CDSVariables_t::Registry, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_RegisterCDSEx().

13.18.2.6 CFE_ES_FormCDSName()

```
void CFE_ES_FormCDSName (
             char * FullCDSName,
             const char * CDSName,
             uint32 ThisAppId )
```

Description

Takes a given CDS Name and combines it with the calling Application's name to make a processor specific name of the form: "AppName.CDSName"

Assumptions, External Events, and Notes:

Note: AppName portion will be truncated to OS_MAX_API_NAME.

Parameters

in	*FullCDSName*	pointer to character buffer of CFE_ES_CDS_MAX_FULL_NAME_LEN size that will be filled with the processor specific CDS Name.
in	*CDSName*	pointer to character string containing the Application's local name for the CDS.
in	*ThisAppId*	the Application ID of the Application making the call.
out	*∗FullCDSName*	processor specific CDS Name of the form "AppName.CDSName".

Return values

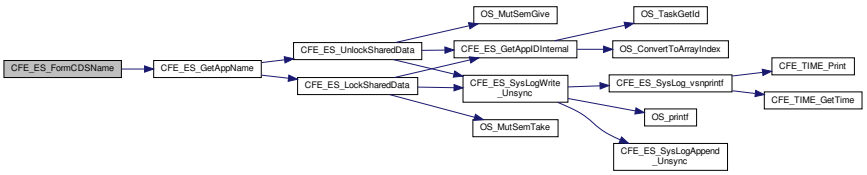
None	

Definition at line 567 of file cfe_es_cds.c.

References CFE_ES_GetAppName(), and OS_MAX_API_NAME.

Referenced by CFE_ES_RegisterCDS().

Here is the call graph for this function:



13.18.2.7 CFE_ES_InitCDSRegistry()

```
int32 CFE_ES_InitCDSRegistry (
            void  )
```

Description

Initializes the data structure used to keep track of CDS blocks and who they belong to.

Assumptions, External Events, and Notes:

None

Return values

CFE_SUCCESS	Operation was performed successfully

Definition at line 473 of file cfe_es_cds.c.

References CDS_REG_SIZE_OFFSET, CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_ES_UpdateCDS↩Registry(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_SUCCESS, CFE_ES_CDSVariables_t::MaxNumRegEntries, CFE_ES_CDS_Reg↩Rec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec↩_t::Size, CFE_ES_CDS_RegRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_InitializeCDS().

Here is the call graph for this function:

13.18.2.8 CFE_ES_InitializeCDS()

```
int32 CFE_ES_InitializeCDS (
            uint32 CDSSize )
```

Description

Stores a fixed pattern at the beginning and end of the CDS memory to tag it for future verification following a reset.

Assumptions, External Events, and Notes:

None

Parameters

in	*CDSSize*	Total size of CDS memory area (in bytes)

Returns

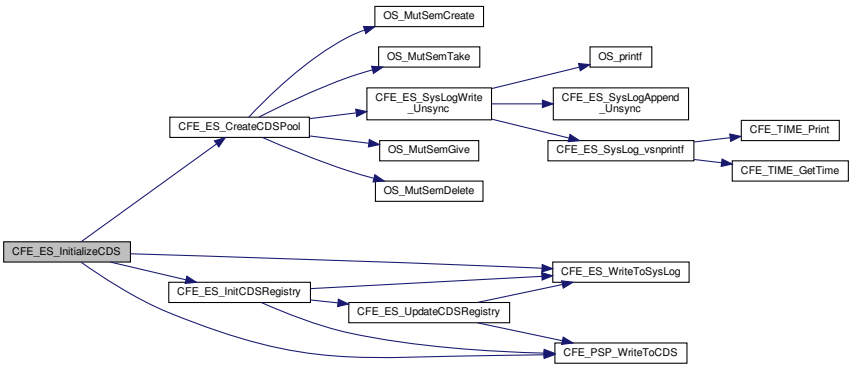
OS_SUCCESS
Any of the return values from CFE_PSP_WriteToCDS
Any of the return values from CFE_ES_CreateCDSPool

Definition at line 381 of file cfe_es_cds.c.

References CDS_POOL_OFFSET, CFE_ES_Global_t::CDSVars, CFE_ES_CreateCDSPool(), CFE_ES_Global, CFE_ES_InitCDSRegistry(), CFE_ES_WriteToSysLog(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_SUCCESS, CFE_ES_CDSVariables_t::MemPoolSize, OS_SUCCESS, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CDS_EarlyInit().

Here is the call graph for this function:



13.18.2.9 CFE_ES_LockCDSRegistry()

```
int32 CFE_ES_LockCDSRegistry (
            void  )
```

Description

Locks the CDS Registry to prevent multiple tasks/threads from modifying it at once.

Assumptions, External Events, and Notes:

None

Return values

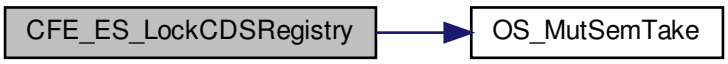
CFE_SUCCESS	Operation was performed successfully

Definition at line 590 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_SUCCESS, OS_MutSemTake(), OS_SUCCESS, and CFE_ES_CDSVariables_t::RegistryMutex.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.18.2.10 CFE_ES_RebuildCDS()

```
int32 CFE_ES_RebuildCDS (
            void  )
```

Description

Scans memory for existing CDS and initializes memory pool and registry settings accordingly

Assumptions, External Events, and Notes:

1. Assumes the validity of the CDS has already been determined

Returns

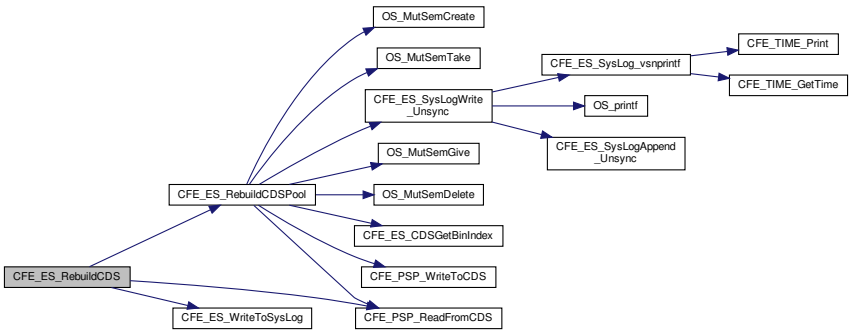
CFE_SUCCESS Operation was performed successfully
Any of the return values from CFE_PSP_ReadFromCDS

Definition at line 697 of file cfe_es_cds.c.

References CDS_REG_OFFSET, CDS_REG_SIZE_OFFSET, CFE_ES_CDSVariables_t::CDSSize, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_INVALID, CFE_ES_Global, CFE_ES_RebuildCDSPool(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_ES_CDSVariables_t::MaxNumRegEntries, CFE_ES_CDSVariables_t::MemPoolSize, CFE_ES_CDSVariables_t::Registry, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CDS_EarlyInit().

Here is the call graph for this function:



13.18.2.11 CFE_ES_RegisterCDSEx()

```
int32 CFE_ES_RegisterCDSEx (
            CFE_ES_CDSHandle_t * HandlePtr,
            int32 BlockSize,
            const char * Name,
            bool CriticalTbl )
```

cFE Core task other function call prototypes

Description

This routine is identical to CFE_ES_RegisterCDS except it identifies the contents of the CDS as a critical table. This is crucial because a critical table CDS must only be deleted by cFE Table Services, not via an ES delete CDS command. Otherwise, Table Services may be out of sync with the contents of the CDS.

Assumptions, External Events, and Notes:

1. This function assumes input parameters are error free and have met size/value restrictions.
2. The calling function is responsible for issuing any event messages associated with errors.

Parameters

in	*HandlePtr*	Pointer Application's variable that will contain the CDS Memory Block Handle.
in	*BlockSize*	The number of bytes needed in the CDS.
in	*Name*	Pointer to character string containing the Application's local name for the CDS.
in	*CriticalTbl*	Indicates whether the CDS is to be used as a Critical Table or not
out	∗*HandlePtr*	The handle of the CDS block that can be used in CFE_ES_CopyToCDS and CFE_ES_RestoreFromCDS.

Returns

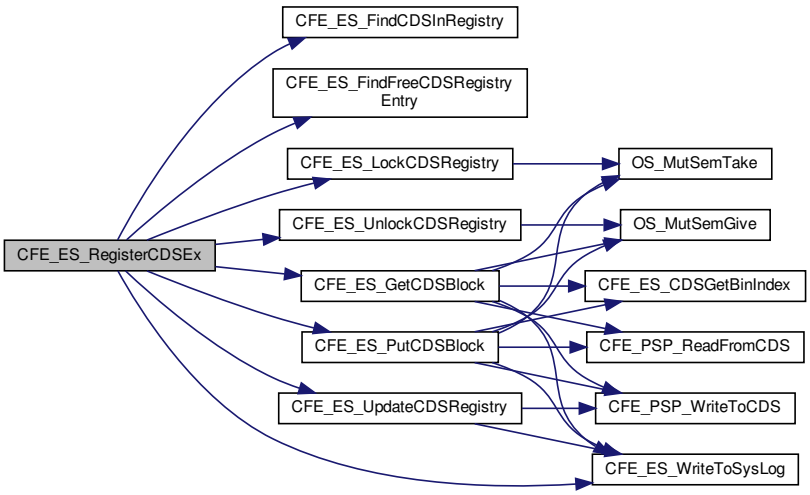
See return codes for CFE_ES_RegisterCDS

Definition at line 230 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_ALREADY_EXISTS, CFE_ES_CDS_MAX_FULL_NAME↩_LEN, CFE_ES_CDS_NOT_FOUND, CFE_ES_CDS_REGISTRY_FULL, CFE_ES_FindCDSInRegistry(), CFE_ES↩_FindFreeCDSRegistryEntry(), CFE_ES_GetCDSBlock(), CFE_ES_Global, CFE_ES_LockCDSRegistry(), CFE_E↩S_PutCDSBlock(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UpdateCDSRegistry(), CFE_ES_WriteToSysLog(), C↩FE_SUCCESS, CFE_ES_CDS_RegRec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, NULL, CFE_ES_CDS↩Variables_t::Registry, CFE_ES_CDS_RegRec_t::Size, CFE_ES_CDS_RegRec_t::Table, and CFE_ES_CDS_Reg↩Rec_t::Taken.

Referenced by CFE_ES_RegisterCDS().

Here is the call graph for this function:



13.18.2.12 CFE_ES_UnlockCDSRegistry()

```
int32 CFE_ES_UnlockCDSRegistry (
            void  )
```

Description

Unlocks CDS Registry to allow other tasks/threads to modify the CDS Registry contents.

Assumptions, External Events, and Notes:

None

Return values

CFE_SUCCESS	Operation was performed successfully

Definition at line 613 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_SUCCESS, OS_MutSemGive(), OS_SUCCESS, and CFE_ES_CDSVariables_t::RegistryMutex.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:

13.18.2.13 CFE_ES_UpdateCDSRegistry()

```
int32 CFE_ES_UpdateCDSRegistry (
            void  )
```

Description

Copies the local working copy of the CDS Registry to the CDS.

Assumptions, External Events, and Notes:

None

Returns

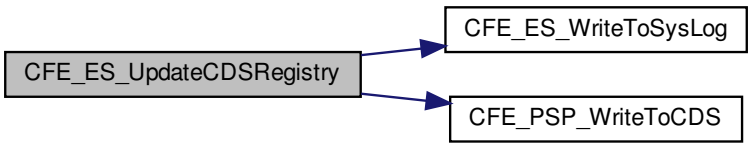
CFE_SUCCESS Operation was performed successfully
Any of the return values from CFE_PSP_WriteToCDS

Definition at line 513 of file cfe_es_cds.c.

References CDS_REG_OFFSET, CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_ES_WriteToSysLog(), CFE_↩ PSP_WriteToCDS(), OS_SUCCESS, and CFE_ES_CDSVariables_t::Registry.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_InitCDSRegistry(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.18.2.14 CFE_ES_ValidateCDS()

```
int32 CFE_ES_ValidateCDS (
            void  )
```

Description

Reads a set of bytes from the beginning and end of the CDS memory area and determines if a fixed pattern is present, thus determining whether the CDS still likely contains valid data or not.

Assumptions, External Events, and Notes:

None

Returns

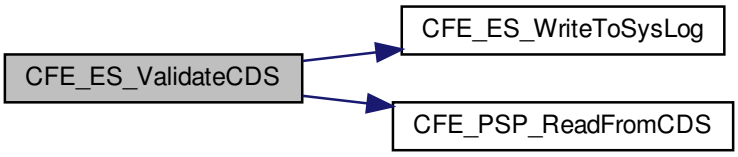
CFE_SUCCESS Operation was performed successfully
CFE_ES_CDS_INVALID The CDS contents are invalid.
Any of the return values from CFE_PSP_ReadFromCDS

Definition at line 328 of file cfe_es_cds.c.

References CFE_ES_CDSVariables_t::CDSSize, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_INVALID, CFE_ES_↩ Global, CFE_ES_WriteToSysLog(), CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_SUCCESS, and CFE↩ _ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CDS_EarlyInit().

Here is the call graph for this function:



13.19 cfe/fsw/cfe-core/src/es/cfe_es_cds.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_es_apps.h"
#include "cfe_platform_cfg.h"
#include "cfe_es.h"
#include "cfe_es_cds_mempool.h"
```

Data Structures

- struct CFE_ES_CDS_RegRec_t
- struct CFE_ES_CDSVariables_t

Registry Mutex Definitions

- #define CFE_ES_CDS_MUT_REG_NAME "CDS_REG_MUT"

  Name of Mutex controlling CDS Registry Access.
- #define CFE_ES_CDS_MUT_REG_VALUE 0

  Initial Value of CDS Registry Access Mutex.
- #define CFE_ES_CDS_NOT_FOUND (uint32)(0xffffffff)
- int32 CFE_ES_CDS_EarlyInit (void)

  Initializes CDS data constructs.
- int32 CFE_ES_UpdateCDSRegistry (void)

  Copies the local version of the CDS Registry to the actual CDS.
- int32 CFE_ES_CDS_ValidateAppID (uint32 *AppIdPtr)

  Validates the Application ID associated with calling Application.
- void CFE_ES_FormCDSName (char *FullCDSName, const char *CDSName, uint32 ThisAppId)

  Creates a Full CDS name from application name and CDS name.
- int32 CFE_ES_FindCDSInRegistry (const char *CDSName)

  Returns the Registry Index for the specified CDS Name.
- int32 CFE_ES_FindFreeCDSRegistryEntry (void)

  Locates a free slot in the CDS Registry.
- int32 CFE_ES_LockCDSRegistry (void)

  Locks access to the CDS Registry.
- int32 CFE_ES_UnlockCDSRegistry (void)

  Unlocks access to the CDS Registry.
- int32 CFE_ES_RebuildCDS (void)

  Rebuilds memory pool for CDS and recovers existing registry.
- int32 CFE_ES_InitCDSRegistry (void)

  Initializes the CDS Registry.
- int32 CFE_ES_ValidateCDS (void)

  Determines whether a CDS currently exists.
- int32 CFE_ES_InitializeCDS (uint32 CDSSize)

  Initializes the contents of the CDS.

13.19.1 Macro Definition Documentation

13.19.1.1 CFE_ES_CDS_MUT_REG_NAME

```
#define CFE_ES_CDS_MUT_REG_NAME "CDS_REG_MUT"
```

Definition at line 57 of file cfe_es_cds.h.

Referenced by CFE_ES_CDS_EarlyInit().

13.19.1.2 CFE_ES_CDS_MUT_REG_VALUE

```
#define CFE_ES_CDS_MUT_REG_VALUE 0
```

Definition at line 58 of file cfe_es_cds.h.

Referenced by CFE_ES_CDS_EarlyInit().

13.19.1.3 CFE_ES_CDS_NOT_FOUND

```
#define CFE_ES_CDS_NOT_FOUND (uint32)(0xffffffff)
```

Definition at line 60 of file cfe_es_cds.h.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_FindCDSInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), and CFE_ES_RegisterCDSEx().

13.19.2 Function Documentation

13.19.2.1 CFE_ES_CDS_EarlyInit()

```
int32 CFE_ES_CDS_EarlyInit (
            void  )
```

Initializes the cFE core module API Library.

Description

Locates and validates any pre-existing CDS memory or initializes the memory as a fresh CDS.

Assumptions, External Events, and Notes:

None

SysLog Messages

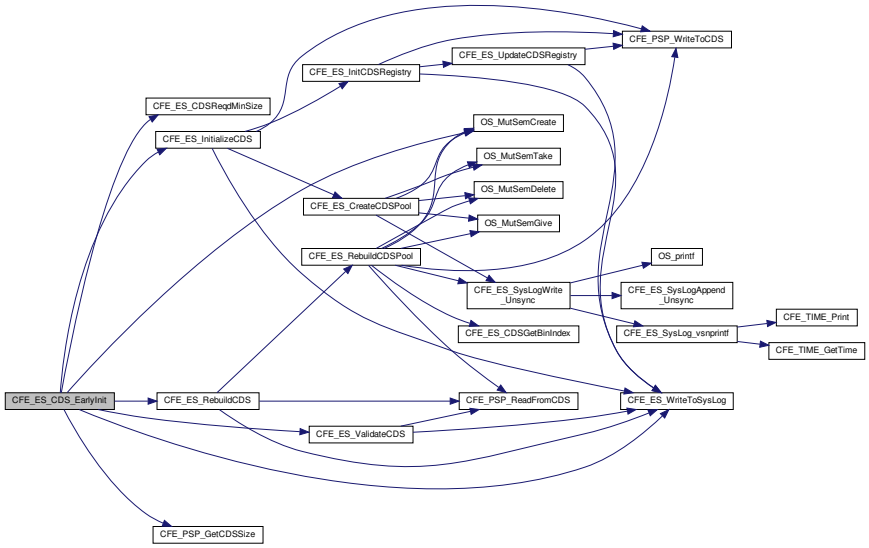
Returns

None

Definition at line 149 of file cfe_es_cds.c.

References CFE_ES_CDSVariables_t::CDSSize, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_INVALID, CFE_ES_↩
CDS_MUT_REG_NAME, CFE_ES_CDS_MUT_REG_VALUE, CFE_ES_CDSReqdMinSize(), CFE_ES_Global, CFE↩
_ES_InitializeCDS(), CFE_ES_RebuildCDS(), CFE_ES_ValidateCDS(), CFE_ES_WriteToSysLog(), CFE_PLATFOR↩
M_ES_CDS_MAX_NUM_ENTRIES, CFE_PSP_GetCDSSize(), CFE_PSP_SUCCESS, CFE_SUCCESS, CFE_ES_↩
CDSVariables_t::MemPoolSize, OS_MutSemCreate(), CFE_ES_CDSVariables_t::RegistryMutex, and CFE_ES_CD↩
SVariables_t::ValidityField.

Here is the call graph for this function:



13.19.2.2 CFE_ES_CDS_ValidateAppID()

```
int32 CFE_ES_CDS_ValidateAppID (
            uint32 * AppIdPtr )
```

Description

Validates Application ID of calling App. Validation consists of ensuring the AppID is between zero and CFE_PLA↩
TFORM_ES_MAX_APPLICATIONS.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppIdPtr*	Pointer to value that will hold AppID on return.
out	∗*AppIdPtr*	The AppID as obtained from CFE_ES_GetAppID

Return values

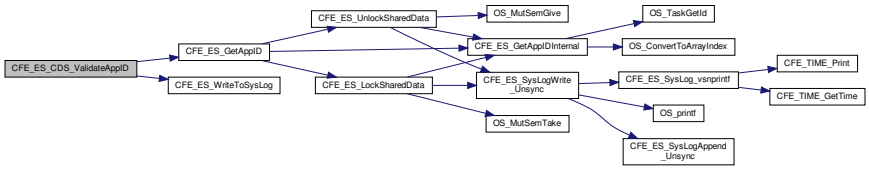
CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.

Definition at line 537 of file cfe_es_cds.c.

References CFE_ES_ERR_APPID, CFE_ES_GetAppID(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX↩
_APPLICATIONS, and CFE_SUCCESS.

Referenced by CFE_ES_RegisterCDS().

Here is the call graph for this function:



13.19.2.3 CFE_ES_FindCDSInRegistry()

```
int32 CFE_ES_FindCDSInRegistry (
            const char * CDSName )
```

Description

Locates given CDS Name in the CDS Registry and returns the appropriate Registry Index.

Assumptions, External Events, and Notes:

None

Parameters

in	*CDSName*	- Pointer to character string containing complete CDS Name (of the format "AppName.CDSName").

Return values

CFE_ES_CDS_NOT_FOUND	or the Index into Registry for Table with specified name

Definition at line 636 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_NOT_FOUND, CFE_ES_Global, CFE_ES_CDSVariables↩
_t::MaxNumRegEntries, CFE_ES_CDS_RegRec_t::Name, CFE_ES_CDSVariables_t::Registry, and CFE_ES_CDS_↩
RegRec_t::Taken.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

13.19.2.4 CFE_ES_FindFreeCDSRegistryEntry()

```
int32 CFE_ES_FindFreeCDSRegistryEntry (
            void  )
```

Description

Locates a free slot in the CDS Registry.

Assumptions, External Events, and Notes:

Note: This function assumes the registry has been locked.

Return values

CFE_ES_CDS_NOT_FOUND	or Index into CDS Registry of unused entry

Definition at line 669 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_NOT_FOUND, CFE_ES_Global, CFE_ES_CDSVariables↩
_t::MaxNumRegEntries, CFE_ES_CDSVariables_t::Registry, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_RegisterCDSEx().

13.19.2.5 CFE_ES_FormCDSName()

```
void CFE_ES_FormCDSName (
            char * FullCDSName,
            const char * CDSName,
            uint32 ThisAppId )
```

Description

Takes a given CDS Name and combines it with the calling Application's name to make a processor specific name of the form: "AppName.CDSName"

Assumptions, External Events, and Notes:

Note: AppName portion will be truncated to OS_MAX_API_NAME.

Parameters

in	*FullCDSName*	pointer to character buffer of CFE_ES_CDS_MAX_FULL_NAME_LEN size that will be filled with the processor specific CDS Name.
in	*CDSName*	pointer to character string containing the Application's local name for the CDS.
in	*ThisAppId*	the Application ID of the Application making the call.
out	**FullCDSName*	processor specific CDS Name of the form "AppName.CDSName".

Return values

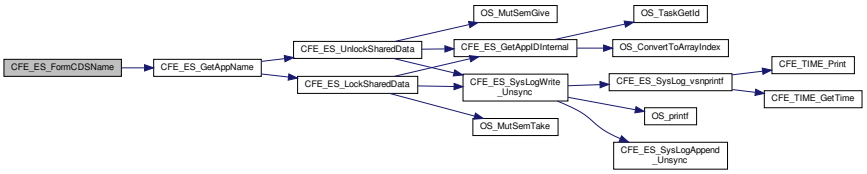
None	

Definition at line 567 of file cfe_es_cds.c.

References CFE_ES_GetAppName(), and OS_MAX_API_NAME.

Referenced by CFE_ES_RegisterCDS().

Here is the call graph for this function:



13.19.2.6 CFE_ES_InitCDSRegistry()

```
int32 CFE_ES_InitCDSRegistry (
            void  )
```

Description

Initializes the data structure used to keep track of CDS blocks and who they belong to.

Assumptions, External Events, and Notes:

None

Return values

CFE_SUCCESS	Operation was performed successfully

Definition at line 473 of file cfe_es_cds.c.

References CDS_REG_SIZE_OFFSET, CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_ES_UpdateCDS↩ Registry(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_SUCCESS, CFE_ES_CDSVariables_t::MaxNumRegEntries, CFE_ES_CDS_Reg↩ Rec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec↩ _t::Size, CFE_ES_CDS_RegRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_InitializeCDS().

Here is the call graph for this function:

13.19.2.7 CFE_ES_InitializeCDS()

```
int32 CFE_ES_InitializeCDS (
            uint32 CDSSize )
```

Description

Stores a fixed pattern at the beginning and end of the CDS memory to tag it for future verification following a reset.

Assumptions, External Events, and Notes:

None

Parameters

in	*CDSSize*	Total size of CDS memory area (in bytes)

Returns

OS_SUCCESS

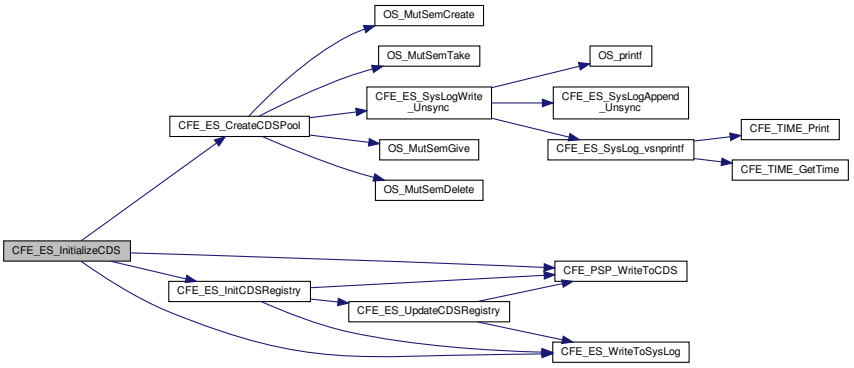
Any of the return values from CFE_PSP_WriteToCDS
Any of the return values from CFE_ES_CreateCDSPool

Definition at line 381 of file cfe_es_cds.c.

References CDS_POOL_OFFSET, CFE_ES_Global_t::CDSVars, CFE_ES_CreateCDSPool(), CFE_ES_Global, CFE_ES_InitCDSRegistry(), CFE_ES_WriteToSysLog(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_SUCCESS, CFE_ES_CDSVariables_t::MemPoolSize, OS_SUCCESS, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CDS_EarlyInit().

Here is the call graph for this function:



13.19.2.8 CFE_ES_LockCDSRegistry()

```
int32 CFE_ES_LockCDSRegistry (
            void  )
```

Description

Locks the CDS Registry to prevent multiple tasks/threads from modifying it at once.

Assumptions, External Events, and Notes:

None

Return values

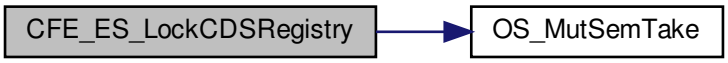
CFE_SUCCESS	Operation was performed successfully

Definition at line 590 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_SUCCESS, OS_MutSemTake(), OS_SUCCESS, and CFE_ES_CDSVariables_t::RegistryMutex.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.19.2.9 CFE_ES_RebuildCDS()

```
int32 CFE_ES_RebuildCDS (
            void  )
```

Description

Scans memory for existing CDS and initializes memory pool and registry settings accordingly

Assumptions, External Events, and Notes:

1. Assumes the validity of the CDS has already been determined

Returns

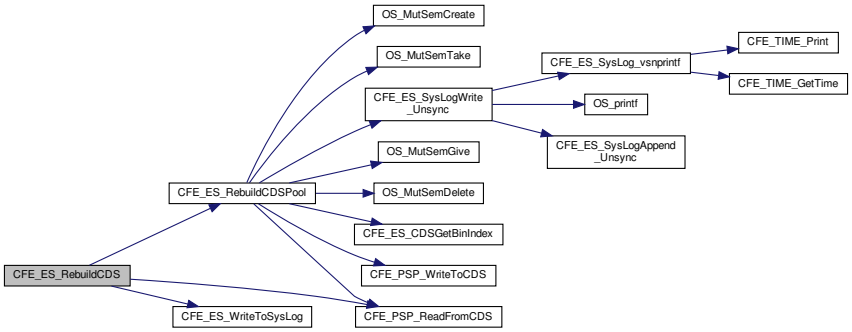
CFE_SUCCESS Operation was performed successfully
Any of the return values from CFE_PSP_ReadFromCDS

Definition at line 697 of file cfe_es_cds.c.

References CDS_REG_OFFSET, CDS_REG_SIZE_OFFSET, CFE_ES_CDSVariables_t::CDSSize, CFE_ES_Global↩_t::CDSVars, CFE_ES_CDS_INVALID, CFE_ES_Global, CFE_ES_RebuildCDSPool(), CFE_ES_WriteToSysLog(), C↩FE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_ES_↩CDSVariables_t::MaxNumRegEntries, CFE_ES_CDSVariables_t::MemPoolSize, CFE_ES_CDSVariables_t::Registry, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CDS_EarlyInit().

Here is the call graph for this function:



13.19.2.10 CFE_ES_UnlockCDSRegistry()

```
int32 CFE_ES_UnlockCDSRegistry (
            void  )
```

Description

Unlocks CDS Registry to allow other tasks/threads to modify the CDS Registry contents.

Assumptions, External Events, and Notes:

None

Return values

CFE_SUCCESS	Operation was performed successfully

Definition at line 613 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_SUCCESS, OS_MutSemGive(), OS_SUCCESS, and CFE_ES_CDSVariables_t::RegistryMutex.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:

13.19.2.11 CFE_ES_UpdateCDSRegistry()

```
int32 CFE_ES_UpdateCDSRegistry (
            void  )
```

Description

Copies the local working copy of the CDS Registry to the CDS.

Assumptions, External Events, and Notes:

None

Returns

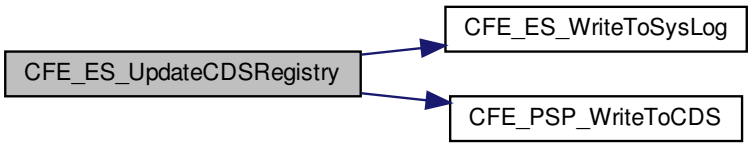
CFE_SUCCESS Operation was performed successfully
Any of the return values from CFE_PSP_WriteToCDS

Definition at line 513 of file cfe_es_cds.c.

References CDS_REG_OFFSET, CFE_ES_Global_t::CDSVars, CFE_ES_Global, CFE_ES_WriteToSysLog(), CFE_↩ PSP_WriteToCDS(), OS_SUCCESS, and CFE_ES_CDSVariables_t::Registry.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_InitCDSRegistry(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.19.2.12 CFE_ES_ValidateCDS()

```
int32 CFE_ES_ValidateCDS (
            void  )
```

Description

Reads a set of bytes from the beginning and end of the CDS memory area and determines if a fixed pattern is present, thus determining whether the CDS still likely contains valid data or not.

Assumptions, External Events, and Notes:

None

Returns

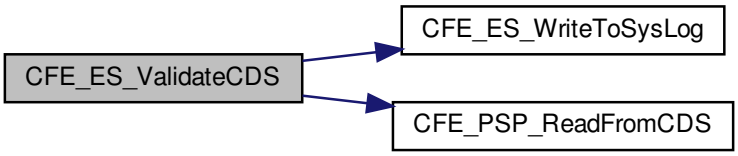
CFE_SUCCESS Operation was performed successfully
CFE_ES_CDS_INVALID The CDS contents are invalid.
Any of the return values from CFE_PSP_ReadFromCDS

Definition at line 328 of file cfe_es_cds.c.

References CFE_ES_CDSVariables_t::CDSSize, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_INVALID, CFE_ES_↩ Global, CFE_ES_WriteToSysLog(), CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_SUCCESS, and CFE↩ _ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CDS_EarlyInit().

Here is the call graph for this function:



13.20 cfe/fsw/cfe-core/src/es/cfe_es_cds_mempool.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es.h"
#include "cfe_psp.h"
#include "cfe_es_cds_mempool.h"
#include "cfe_es_global.h"
#include "cfe_es_log.h"
#include <stdio.h>
```

Macros

- #define CFE_ES_CDS_CHECK_PATTERN 0x5a5a
- #define CFE_ES_CDS_BLOCK_USED 0xaaaa
- #define CFE_ES_CDS_BLOCK_UNUSED 0xdddd

Functions

- int32 CFE_ES_CDSGetBinIndex (uint32 DesiredSize)
- int32 CFE_ES_CreateCDSPool (uint32 CDSPoolSize, uint32 StartOffset)

  Creates a CDS memory pool from scratch.
- int32 CFE_ES_RebuildCDSPool (uint32 CDSPoolSize, uint32 StartOffset)
- int32 CFE_ES_GetCDSBlock (CFE_ES_CDSBlockHandle_t *BlockHandle, uint32 BlockSize)
- int32 CFE_ES_PutCDSBlock (CFE_ES_CDSBlockHandle_t BlockHandle)
- int32 CFE_ES_CDSBlockWrite (CFE_ES_CDSBlockHandle_t BlockHandle, void *DataToWrite)
- int32 CFE_ES_CDSBlockRead (void *DataRead, CFE_ES_CDSBlockHandle_t BlockHandle)
- uint32 CFE_ES_CDSReqdMinSize (uint32 MaxNumBlocksToSupport)

Variables

- CFE_ES_CDSPool_t CFE_ES_CDSMemPool
- CFE_ES_CDSBlockDesc_t CFE_ES_CDSBlockDesc
- uint32 CFE_ES_CDSMemPoolDefSize [CFE_ES_CDS_NUM_BLOCK_SIZES]

13.20.1 Macro Definition Documentation

13.20.1.1 CFE_ES_CDS_BLOCK_UNUSED

```
#define CFE_ES_CDS_BLOCK_UNUSED 0xdddd
```

Definition at line 51 of file cfe_es_cds_mempool.c.

Referenced by CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

13.20.1.2 CFE_ES_CDS_BLOCK_USED

```
#define CFE_ES_CDS_BLOCK_USED 0xaaaa
```

Definition at line 50 of file cfe_es_cds_mempool.c.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCD↩SBlock(), and CFE_ES_RebuildCDSPool().

13.20.1.3 CFE_ES_CDS_CHECK_PATTERN

```
#define CFE_ES_CDS_CHECK_PATTERN 0x5a5a
```

Definition at line 49 of file cfe_es_cds_mempool.c.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

13.20.2 Function Documentation

13.20.2.1 CFE_ES_CDSBlockRead()

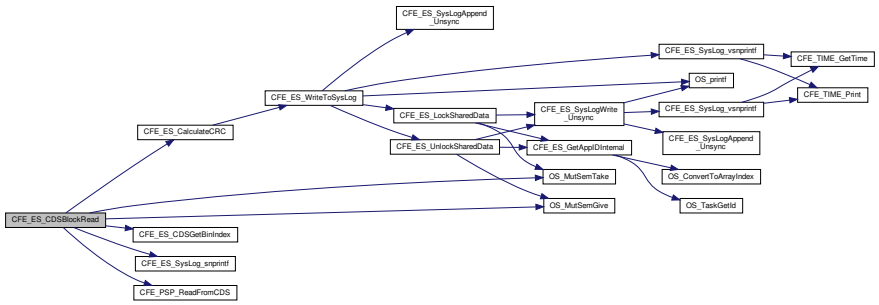
```
int32 CFE_ES_CDSBlockRead (
            void * DataRead,
            CFE_ES_CDSBlockHandle_t BlockHandle )
```

Definition at line 581 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_Global_t::CDSVars, CFE_ES_CalculateCRC(), CFE_ES_CDS_BLOCK_CRC_ERR, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDSGetBinIndex(), CFE_ES_ERR_MEM_HANDLE, CFE_ES_Global, CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_MISSION_ES_DEFAULT_CRC, CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_SUCCESS, CFE_ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSBlockDesc_t::CRC, CFE_ES_CDSPool_t::End, CFE_ES_CDSPool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSBlockDesc_t::SizeUsed, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_RestoreFromCDS().

Here is the call graph for this function:



13.20.2.2 CFE_ES_CDSBlockWrite()

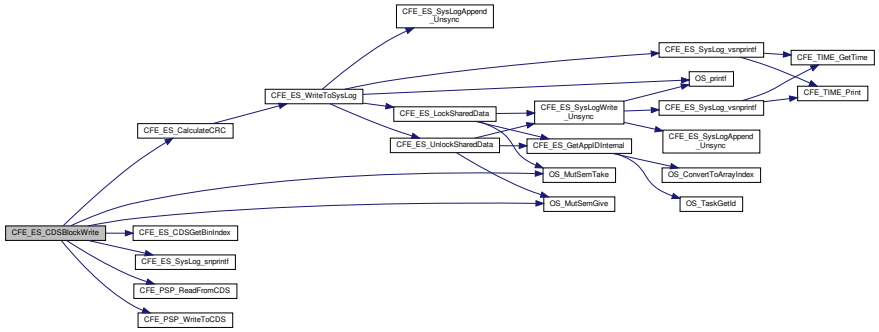
```
int32 CFE_ES_CDSBlockWrite (
            CFE_ES_CDSBlockHandle_t BlockHandle,
            void * DataToWrite )
```

Definition at line 480 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_Global_t::↩ CDSVars, CFE_ES_CalculateCRC(), CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES↩ _CDSGetBinIndex(), CFE_ES_ERR_MEM_HANDLE, CFE_ES_Global, CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE↩ _ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_MISSION_ES_DEFAULT_CRC, CFE_PSP_ReadFrom↩ CDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_SUCCESS, CFE_ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSBlockDesc_t::CRC, CFE_ES_CDSPool_t::End, CFE_ES_CDS↩ Pool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSBlock↩ Desc_t::SizeUsed, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CopyToCDS().

Here is the call graph for this function:



13.20.2.3 CFE_ES_CDSGetBinIndex()

```
int32 CFE_ES_CDSGetBinIndex (
            uint32 DesiredSize )
```

Definition at line 453 of file cfe_es_cds_mempool.c.

References CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSBlockSizeDesc_t::MaxSize, and CFE_ES_CDS↩ Pool_t::SizeDesc.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCD↩ SBlock(), and CFE_ES_RebuildCDSPool().

13.20.2.4 CFE_ES_CDSReqdMinSize()

```
uint32 CFE_ES_CDSReqdMinSize (
            uint32 MaxNumBlocksToSupport )
```

Definition at line 680 of file cfe_es_cds_mempool.c.

References CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSMemPoolDefSize, and CFE_ES_CDSPool_t::MinBlockSize.

Referenced by CFE_ES_CDS_EarlyInit().

13.20.2.5 CFE_ES_CreateCDSPool()

```
int32 CFE_ES_CreateCDSPool (
            uint32 CDSPoolSize,
            uint32 StartOffset )
```

Description

Creates a memory pool of the specified size starting at the specified offset into the CDS memory.

Assumptions, External Events, and Notes:

None

Returns

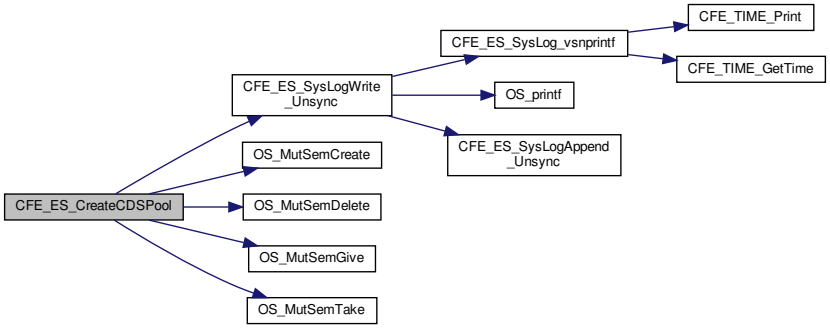
CFE_SUCCESS Operation was performed successfully

Definition at line 106 of file cfe_es_cds_mempool.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSMemPoolDefSize, CFE_ES_SysLogWrite_Unsync(), CFE_SUCCESS, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSPool_t::Current, CFE_ES_CDSPool_t::End, CFE_ES_CDSBlockSizeDesc_t::MaxSize, CFE_ES_CDSPool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockSizeDesc_t::NumCreated, OS_MutSemCreate(), OS_MutSemDelete(), OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSPool_t::RequestCntr, CFE_ES_CDSPool_t::Size, CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSPool_t::SizeIndex, CFE_ES_CDSPool_t::Start, and CFE_ES_CDSBlockSizeDesc_t::Top.

Referenced by CFE_ES_InitializeCDS().

Here is the call graph for this function:



13.20.2.6 CFE_ES_GetCDSBlock()

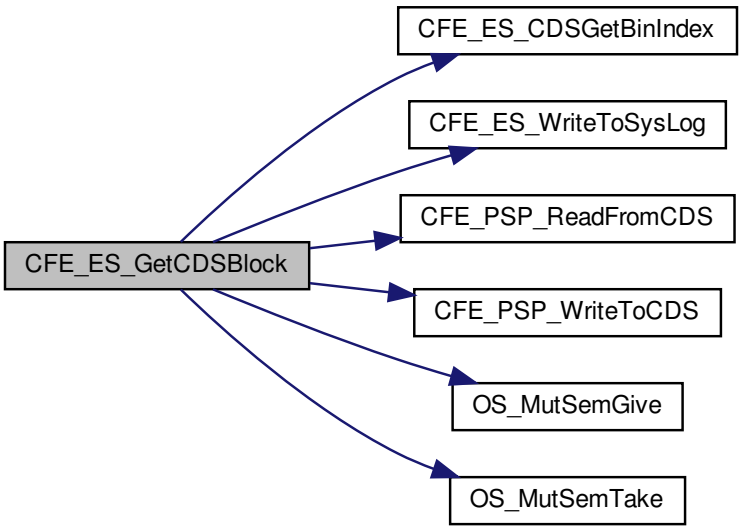
```
int32 CFE_ES_GetCDSBlock (
            CFE_ES_CDSBlockHandle_t * BlockHandle,
            uint32 BlockSize )
```

Definition at line 280 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_CDS_A↩CCESS_ERROR, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDSGetBinIndex(), CFE_ES_ERR_MEM_BLOCK_SIZE, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE, CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_ES_CDSBlockDesc_t::Check↩Bits, CFE_ES_CDSBlockDesc_t::CRC, CFE_ES_CDSPool_t::Current, CFE_ES_CDSPool_t::End, CFE_ES_CDS↩BlockSizeDesc_t::MaxSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockDesc_t::Next, CFE_ES_CDSBlock↩SizeDesc_t::NumCreated, OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSPool_t::RequestCntr, CFE_ES_C↩DSPool_t::SizeDesc, CFE_ES_CDSBlockDesc_t::SizeUsed, and CFE_ES_CDSBlockSizeDesc_t::Top.

Referenced by CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.20.2.7 CFE_ES_PutCDSBlock()

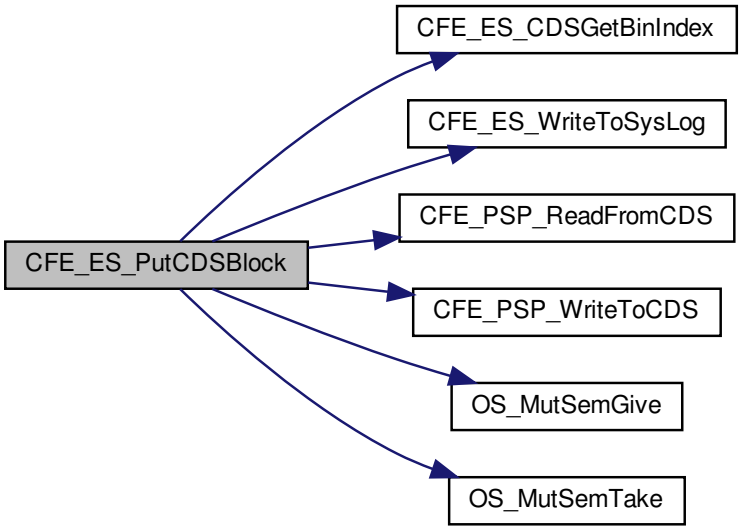
```
int32 CFE_ES_PutCDSBlock (
            CFE_ES_CDSBlockHandle_t BlockHandle )
```

Definition at line 380 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_Global_↩ t::CDSVars, CFE_ES_CDS_ACCESS_ERROR, CFE_ES_CDS_BLOCK_UNUSED, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDSGetBinIndex(), CFE_ES_ERR_MEM_HANDLE, CFE_ES_Global, CFE_ES_WriteToSysLog(), CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_↩ ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSPool_t::End, CFE_ES_CDS↩ Pool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockDesc_t::Next, OS_MutSemGive(), OS_↩ MutSemTake(), CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSBlockSizeDesc_t::Top, and CFE_ES_CDSVariables↩ _t::ValidityField.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.20.2.8 CFE_ES_RebuildCDSPool()

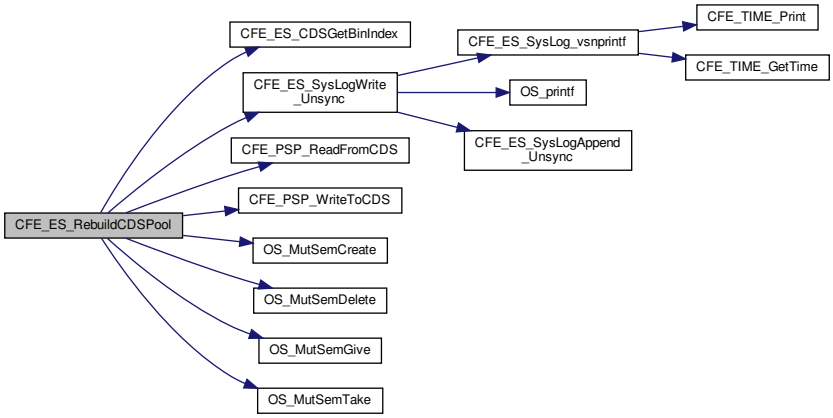
```
int32 CFE_ES_RebuildCDSPool (
            uint32 CDSPoolSize,
            uint32 StartOffset )
```

Definition at line 163 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_BAD_ARGUMENT, CFE_ES_CDS_ACCESS_ERROR, CFE_ES_CDS_BLOCK_UNUSED, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSGetBinIndex(), CFE_ES_CDSMemPoolDefSize, CFE_ES_SysLogWrite_Unsync(), CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSPool_t::Current, CFE_ES_CDSPool_t::End, CFE_ES_CDSBlockSizeDesc_t::MaxSize, CFE_ES_CDSPool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockDesc_t::Next, CFE_ES_CDSBlockSizeDesc_t::NumCreated, OS_MutSemCreate(), OS_MutSemDelete(), OS_MutSemGive(), OS_MutSemTake(), OS_SUCCESS, CFE_ES_CDSPool_t::RequestCntr, CFE_ES_CDSPool_t::Size, CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSPool_t::SizeIndex, CFE_ES_CDSBlockDesc_t::SizeUsed, CFE_ES_CDSPool_t::Start, and CFE_ES_CDSBlockSizeDesc_t::Top.

Referenced by CFE_ES_RebuildCDS().

Here is the call graph for this function:



13.20.3 Variable Documentation

13.20.3.1 CFE_ES_CDSBlockDesc

```
CFE_ES_CDSBlockDesc_t CFE_ES_CDSBlockDesc
```

Definition at line 64 of file cfe_es_cds_mempool.c.

13.20.3.2 CFE_ES_CDSMemPool

```
CFE_ES_CDSPool_t CFE_ES_CDSMemPool
```

Definition at line 63 of file cfe_es_cds_mempool.c.

13.20.3.3 CFE_ES_CDSMemPoolDefSize

```
uint32 CFE_ES_CDSMemPoolDefSize[CFE_ES_CDS_NUM_BLOCK_SIZES]
```

Initial value:

```
=
{
    CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_15,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_14,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_13,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_12,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_11,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_10,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_09,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_08,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_07,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_06,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02,
    CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01
}
```

Definition at line 66 of file cfe_es_cds_mempool.c.

Referenced by CFE_ES_CDSReqdMinSize(), CFE_ES_CreateCDSPool(), and CFE_ES_RebuildCDSPool().

13.21 cfe/fsw/cfe-core/src/es/cfe_es_cds_mempool.h File Reference

```
#include "private/cfe_private.h"
```

Data Structures

- struct CFE_ES_CDSBlockDesc_t
- struct CFE_ES_CDSBlockSizeDesc_t
- struct CFE_ES_CDSPool_t

Macros

- #define CFE_ES_CDS_NUM_BLOCK_SIZES 17

Typedefs

- typedef uint32 CFE_ES_CDSBlockHandle_t

Functions

- int32 CFE_ES_CreateCDSPool (uint32 CDSPoolSize, uint32 StartOffset)

  Creates a CDS memory pool from scratch.
- int32 CFE_ES_RebuildCDSPool (uint32 CDSPoolSize, uint32 StartOffset)
- int32 CFE_ES_GetCDSBlock (CFE_ES_CDSBlockHandle_t *BlockHandle, uint32 BlockSize)
- int32 CFE_ES_PutCDSBlock (CFE_ES_CDSBlockHandle_t BlockHandle)
- int32 CFE_ES_CDSBlockWrite (CFE_ES_CDSBlockHandle_t BlockHandle, void *DataToWrite)
- int32 CFE_ES_CDSBlockRead (void *DataRead, CFE_ES_CDSBlockHandle_t BlockHandle)
- uint32 CFE_ES_CDSReqdMinSize (uint32 MaxNumBlocksToSupport)

Variables

- CFE_ES_CDSPool_t CFE_ES_CDSMemPool
- CFE_ES_CDSBlockDesc_t CFE_ES_CDSBlockDesc

13.21.1 Macro Definition Documentation

13.21.1.1 CFE_ES_CDS_NUM_BLOCK_SIZES

```
#define CFE_ES_CDS_NUM_BLOCK_SIZES 17
```

Definition at line 49 of file cfe_es_cds_mempool.h.

Referenced by CFE_ES_CDSGetBinIndex(), CFE_ES_CDSReqdMinSize(), CFE_ES_CreateCDSPool(), and CFE_E↩S_RebuildCDSPool().

13.21.2 Typedef Documentation

13.21.2.1 CFE_ES_CDSBlockHandle_t

```
typedef uint32 CFE_ES_CDSBlockHandle_t
```

Definition at line 55 of file cfe_es_cds_mempool.h.

13.21.3 Function Documentation

13.21.3.1 CFE_ES_CDSBlockRead()

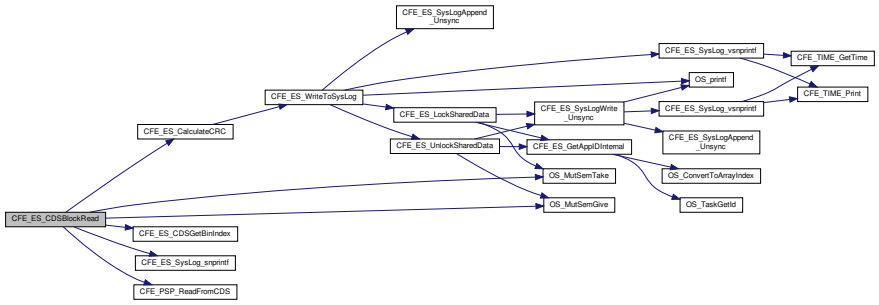
```
int32 CFE_ES_CDSBlockRead (
            void * DataRead,
            CFE_ES_CDSBlockHandle_t BlockHandle )
```

Definition at line 581 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_Global_t::↩
CDSVars, CFE_ES_CalculateCRC(), CFE_ES_CDS_BLOCK_CRC_ERR, CFE_ES_CDS_BLOCK_USED, CFE_ES↩
_CDS_CHECK_PATTERN, CFE_ES_CDSGetBinIndex(), CFE_ES_ERR_MEM_HANDLE, CFE_ES_Global, CFE_↩
ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_MISSION_ES_D↩
EFAULT_CRC, CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_SUCCESS, CFE_ES_CDSBlockDesc_t::↩
CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSBlockDesc_t::CRC, CFE_ES_CDSPool_t::End, CFE↩
_ES_CDSPool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, OS_MutSemGive(), OS_MutSemTake(), CFE_ES_↩
CDSBlockDesc_t::SizeUsed, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_RestoreFromCDS().

Here is the call graph for this function:



13.21.3.2 CFE_ES_CDSBlockWrite()

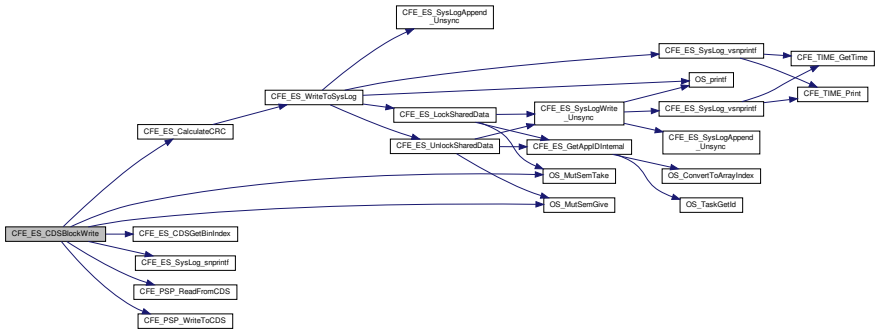
```
int32 CFE_ES_CDSBlockWrite (
            CFE_ES_CDSBlockHandle_t BlockHandle,
            void * DataToWrite )
```

Definition at line 480 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_Global_t::↩
CDSVars, CFE_ES_CalculateCRC(), CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES↩
_CDSGetBinIndex(), CFE_ES_ERR_MEM_HANDLE, CFE_ES_Global, CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE↩
_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_MISSION_ES_DEFAULT_CRC, CFE_PSP_ReadFrom↩
CDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_SUCCESS, CFE_ES_CDSBlockDesc_t::CheckBits,
CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSBlockDesc_t::CRC, CFE_ES_CDSPool_t::End, CFE_ES_CDS↩
Pool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSBlock↩
Desc_t::SizeUsed, and CFE_ES_CDSVariables_t::ValidityField.

Referenced by CFE_ES_CopyToCDS().

Here is the call graph for this function:



13.21.3.3 CFE_ES_CDSReqdMinSize()

```
uint32 CFE_ES_CDSReqdMinSize (
            uint32 MaxNumBlocksToSupport )
```

Definition at line 680 of file cfe_es_cds_mempool.c.

References CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSMemPoolDefSize, and CFE_ES_CDSPool_t::Min↩ BlockSize.

Referenced by CFE_ES_CDS_EarlyInit().

13.21.3.4 CFE_ES_CreateCDSPool()

```
int32 CFE_ES_CreateCDSPool (
            uint32 CDSPoolSize,
            uint32 StartOffset )
```

Description

Creates a memory pool of the specified size starting at the specified offset into the CDS memory.

Assumptions, External Events, and Notes:

None

Returns

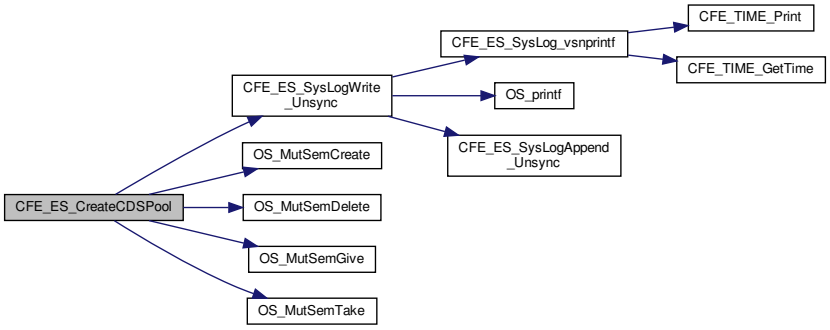
CFE_SUCCESS Operation was performed successfully

Definition at line 106 of file cfe_es_cds_mempool.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSMemPoolDefSize, CFE_ES_SysLogWrite_Unsync(), CFE_SUCCESS, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSPool_t::Current, CFE_ES_CDSPool_t::End, CFE_ES_CDSBlockSizeDesc_t::MaxSize, CFE_ES_CDSPool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockSizeDesc_t::NumCreated, OS_MutSemCreate(), OS_MutSemDelete(), OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSPool_t::RequestCntr, CFE_ES_CDSPool_t::Size, CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSPool_t::SizeIndex, CFE_ES_CDSPool_t::Start, and CFE_ES_CDSBlockSizeDesc_t::Top.

Referenced by CFE_ES_InitializeCDS().

Here is the call graph for this function:



13.21.3.5 CFE_ES_GetCDSBlock()

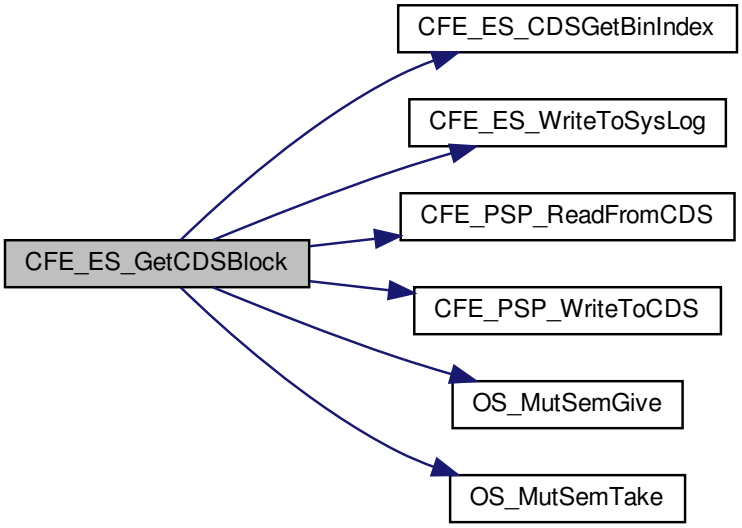
```
int32 CFE_ES_GetCDSBlock (
            CFE_ES_CDSBlockHandle_t * BlockHandle,
            uint32 BlockSize )
```

Definition at line 280 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_CDS_ACCESS_ERROR, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDSGetBinIndex(), CFE_ES_ERR_MEM_BLOCK_SIZE, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_CDS_MAX_BLOCK_SIZE, CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSBlockDesc_t::CRC, CFE_ES_CDSPool_t::Current, CFE_ES_CDSPool_t::End, CFE_ES_CDSBlockSizeDesc_t::MaxSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockDesc_t::Next, CFE_ES_CDSBlockSizeDesc_t::NumCreated, OS_MutSemGive(), OS_MutSemTake(), CFE_ES_CDSPool_t::RequestCntr, CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSBlockDesc_t::SizeUsed, and CFE_ES_CDSBlockSizeDesc_t::Top.

Referenced by CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.21.3.6 CFE_ES_PutCDSBlock()

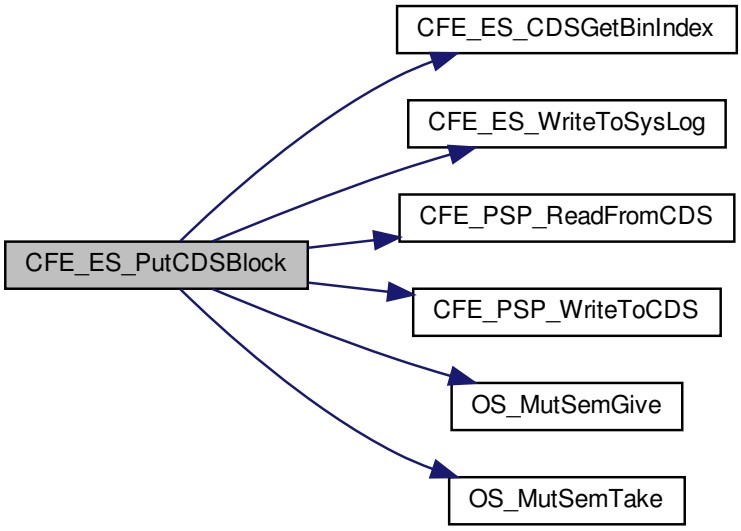
```
int32 CFE_ES_PutCDSBlock (
            CFE_ES_CDSBlockHandle_t BlockHandle )
```

Definition at line 380 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_Global_↩ t::CDSVars, CFE_ES_CDS_ACCESS_ERROR, CFE_ES_CDS_BLOCK_UNUSED, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDSGetBinIndex(), CFE_ES_ERR_MEM_HANDLE, CFE_ES_Global, CFE_ES_WriteToSysLog(), CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_↩ ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSPool_t::End, CFE_ES_CDS↩ Pool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockDesc_t::Next, OS_MutSemGive(), OS_↩ MutSemTake(), CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSBlockSizeDesc_t::Top, and CFE_ES_CDSVariables↩ _t::ValidityField.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_RegisterCDSEx().

Here is the call graph for this function:



13.21.3.7 CFE_ES_RebuildCDSPool()

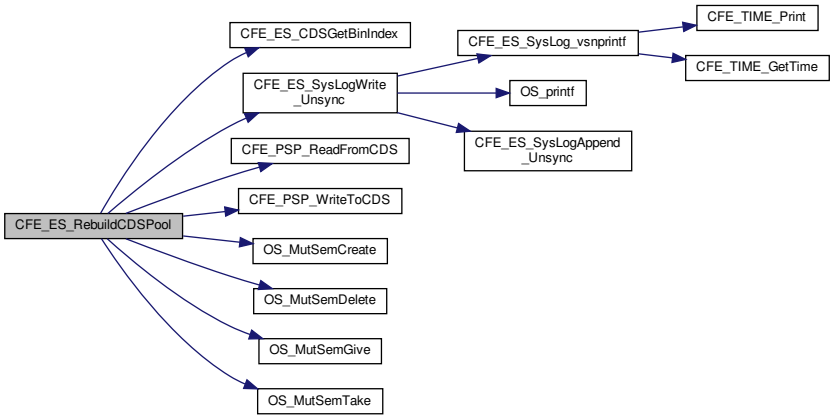
```
int32 CFE_ES_RebuildCDSPool (
            uint32 CDSPoolSize,
            uint32 StartOffset )
```

Definition at line 163 of file cfe_es_cds_mempool.c.

References CFE_ES_CDSBlockDesc_t::ActualSize, CFE_ES_CDSBlockDesc_t::AllocatedFlag, CFE_ES_BAD_ARGUMENT, CFE_ES_CDS_ACCESS_ERROR, CFE_ES_CDS_BLOCK_UNUSED, CFE_ES_CDS_BLOCK_USED, CFE_ES_CDS_CHECK_PATTERN, CFE_ES_CDS_NUM_BLOCK_SIZES, CFE_ES_CDSGetBinIndex(), CFE_ES_CDSMemPoolDefSize, CFE_ES_SysLogWrite_Unsync(), CFE_PSP_ReadFromCDS(), CFE_PSP_SUCCESS, CFE_PSP_WriteToCDS(), CFE_ES_CDSBlockDesc_t::CheckBits, CFE_ES_CDSPool_t::CheckErrCntr, CFE_ES_CDSPool_t::Current, CFE_ES_CDSPool_t::End, CFE_ES_CDSBlockSizeDesc_t::MaxSize, CFE_ES_CDSPool_t::MinBlockSize, CFE_ES_CDSPool_t::MutexId, CFE_ES_CDSBlockDesc_t::Next, CFE_ES_CDSBlockSizeDesc_t::NumCreated, OS_MutSemCreate(), OS_MutSemDelete(), OS_MutSemGive(), OS_MutSemTake(), OS_SUCCESS, CFE_ES_CDSPool_t::RequestCntr, CFE_ES_CDSPool_t::Size, CFE_ES_CDSPool_t::SizeDesc, CFE_ES_CDSPool_t::SizeIndex, CFE_ES_CDSBlockDesc_t::SizeUsed, CFE_ES_CDSPool_t::Start, and CFE_ES_CDSBlockSizeDesc_t::Top.

Referenced by CFE_ES_RebuildCDS().

Here is the call graph for this function:



13.21.4 Variable Documentation

13.21.4.1 CFE_ES_CDSBlockDesc

```
CFE_ES_CDSBlockDesc_t CFE_ES_CDSBlockDesc
```

Definition at line 64 of file cfe_es_cds_mempool.c.

13.21.4.2 CFE_ES_CDSMemPool

```
CFE_ES_CDSPool_t CFE_ES_CDSMemPool
```

Definition at line 63 of file cfe_es_cds_mempool.c.

13.22 cfe/fsw/cfe-core/src/es/cfe_es_erlog.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es.h"
#include "cfe_es_apps.h"
#include "cfe_es_global.h"
#include "cfe_es_log.h"
#include "cfe_psp.h"
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
```

Functions

- CompileTimeAssert (sizeof(CFE_ES_ResetDataPtr->ERLog[0].Context)==CFE_PLATFORM_ES_ER_LOG_↩ MAX_CONTEXT_SIZE, CfeEsErLogContextSizeError)
- int32 CFE_ES_WriteToERLog (uint32 EntryType, uint32 ResetType, uint32 ResetSubtype, const char ∗Description, const uint32 ∗Context, uint32 ContextSize)

13.22.1 Function Documentation

13.22.1.1 CFE_ES_WriteToERLog()

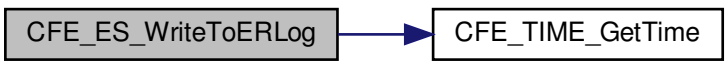
```
int32 CFE_ES_WriteToERLog (
            uint32 EntryType,
            uint32 ResetType,
            uint32 ResetSubtype,
            const char * Description,
            const uint32 * Context,
            uint32 ContextSize )
```

Definition at line 68 of file cfe_es_erlog.c.

References CFE_ES_ResetVariables_t::BootSource, CFE_ES_ERLog_t::BootSource, CFE_ES_Global, CFE_ES↩ _ResetDataPtr, CFE_PLATFORM_ES_ER_LOG_ENTRIES, CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SI↩ ZE, CFE_SUCCESS, CFE_TIME_GetTime(), CFE_ES_ERLog_t::Context, CFE_ES_ERLog_t::ContextSize, CFE_E↩ S_ERLog_t::DebugVars, CFE_ES_Global_t::DebugVars, CFE_ES_ERLog_t::Description, CFE_ES_ResetData_t::E↩ RLog, CFE_ES_ResetData_t::ERLogEntries, CFE_ES_ResetData_t::ERLogIndex, CFE_ES_ERLog_t::LogEntryType, CFE_ES_ResetVariables_t::MaxProcessorResetCount, CFE_ES_ERLog_t::MaxProcessorResetCount, NULL, CFE_↩ ES_ResetVariables_t::ProcessorResetCount, CFE_ES_ERLog_t::ProcessorResetCount, CFE_ES_ERLog_t::Reset↩ Subtype, CFE_ES_ERLog_t::ResetType, CFE_ES_ResetData_t::ResetVars, and CFE_ES_ERLog_t::TimeCode.

Referenced by CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), and CFE_ES_SetupResetVariables().

Here is the call graph for this function:



13.22.1.2 CompileTimeAssert()

```
CompileTimeAssert (
            sizeof(CFE_ES_ResetDataPtr->ERLog[0].Context)  = =CFE_PLATFORM_ES_ER_LOG_MAX_CONTEX↩
T_SIZE,
            CfeEsErLogContextSizeError  )
```

13.23 cfe/fsw/cfe-core/src/es/cfe_es_global.h File Reference

```
#include "osapi.h"
#include "private/cfe_private.h"
#include "private/cfe_es_resetdata_typedef.h"
#include "cfe_es.h"
#include "cfe_es_apps.h"
#include "cfe_es_cds.h"
#include "cfe_es_perf.h"
#include "cfe_time.h"
#include "cfe_platform_cfg.h"
#include "cfe_evs.h"
#include "cfe_psp.h"
```

Data Structures

- struct CFE_ES_GenCounterRecord_t
- struct CFE_ES_Global_t

Functions

- int32 CFE_ES_GetAppIDInternal (uint32 *AppIdPtr)
- void CFE_ES_LockSharedData (const char *FunctionName, int32 LineNumber)
- void CFE_ES_UnlockSharedData (const char *FunctionName, int32 LineNumber)

Variables

- CFE_ES_Global_t CFE_ES_Global
- CFE_ES_ResetData_t * CFE_ES_ResetDataPtr

13.23.1 Function Documentation

13.23.1.1 CFE_ES_GetAppIDInternal()

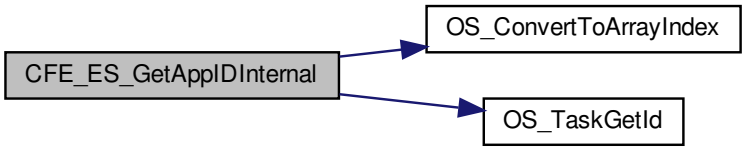
```
int32 CFE_ES_GetAppIDInternal (
            uint32 * AppIdPtr )
```

Definition at line 1728 of file cfe_es_api.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_SUCCESS, OS_Convert↩ ToArrayIndex(), OS_SUCCESS, OS_TaskGetId(), CFE_ES_TaskRecord_t::RecordUsed, and CFE_ES_Global_t::↩ TaskTable.

Referenced by CFE_ES_CreateChildTask(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_GetAppID(), C↩ FE_ES_LockSharedData(), CFE_ES_RunLoop(), CFE_ES_UnlockSharedData(), and CFE_ES_WaitForSystemState().

Here is the call graph for this function:



13.23.1.2 CFE_ES_LockSharedData()

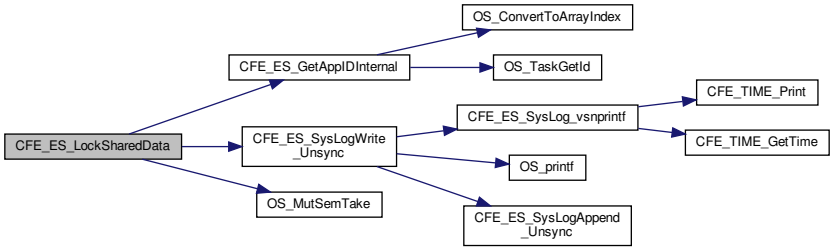
```
void CFE_ES_LockSharedData (
            const char * FunctionName,
            int32 LineNumber )
```

Definition at line 1775 of file cfe_es_api.c.

References CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_SysLogWrite_Unsync(), OS_MutSemTake(), O↩ S_SUCCESS, and CFE_ES_Global_t::SharedDataMutex.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_ClearSyslogCmd(), CFE_ES_CreateChild↩ Task(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_ExitApp(), CFE↩ _ES_ExitChildTask(), CFE_ES_GetAppID(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_↩ ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_ES_LoadLibrary(), CFE_ES_MainTaskSyncDelay(), CFE_ES_↩ RegisterApp(), CFE_ES_RegisterChildTask(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_SysLogDump(), CFE_ES_WaitForSystemState(), and CFE_ES_WriteToSysLog().

Here is the call graph for this function:



13.23.1.3 CFE_ES_UnlockSharedData()

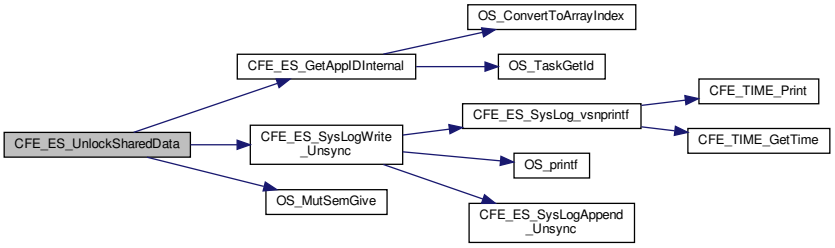
```
void CFE_ES_UnlockSharedData (
            const char * FunctionName,
            int32 LineNumber )
```

Definition at line 1813 of file cfe_es_api.c.

References CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_SysLogWrite_Unsync(), OS_MutSemGive(), OS_SUCCESS, and CFE_ES_Global_t::SharedDataMutex.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_ClearSyslogCmd(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_GetAppID(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetTaskInfo(), CFE_ES_LoadLibrary(), CFE_ES_MainTaskSyncDelay(), CFE_ES_RegisterApp(), CFE_ES_RegisterChildTask(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_SysLogDump(), CFE_ES_WaitForSystemState(), and CFE_ES_WriteToSysLog().

Here is the call graph for this function:



13.23.2 Variable Documentation

13.23.2.1 CFE_ES_Global

```
CFE_ES_Global_t CFE_ES_Global
```

Definition at line 68 of file cfe_es_start.c.

Referenced by CFE_ES_AppCreate(), CFE_ES_CDS_EarlyInit(), CFE_ES_CDSBlockRead(), CFE_ES_CDSBlock↩ Write(), CFE_ES_CleanUpApp(), CFE_ES_CleanupTaskResources(), CFE_ES_CopyToCDS(), CFE_ES_Create↩ ChildTask(), CFE_ES_DeleteApp(), CFE_ES_DeleteCDS(), CFE_ES_DeleteChildTask(), CFE_ES_DeleteGen↩ Counter(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_FindCD↩ SInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppIDInternal(), CFE_ES_GetAppInfo(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetGenCount(), CF↩ E_ES_GetGenCounterIDByName(), CFE_ES_GetTaskInfo(), CFE_ES_HousekeepingCmd(), CFE_ES_Increment↩ GenCounter(), CFE_ES_IncrementTaskCounter(), CFE_ES_InitCDSRegistry(), CFE_ES_InitializeCDS(), CFE_ES↩ _ListApplications(), CFE_ES_ListTasks(), CFE_ES_LoadLibrary(), CFE_ES_LockCDSRegistry(), CFE_ES_Lock↩ SharedData(), CFE_ES_ProcessControlRequest(), CFE_ES_ProcessCoreException(), CFE_ES_PutCDSBlock(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_RebuildCDS(), CFE_ES_RegisterCDS(), C↩ FE_ES_RegisterCDSEx(), CFE_ES_RegisterGenCounter(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CF↩ E_ES_RestoreFromCDS(), CFE_ES_RunLoop(), CFE_ES_ScanAppTable(), CFE_ES_SetAppState(), CFE_ES↩ _SetGenCount(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UnlockSharedData(), CFE_ES_UpdateCDSRegistry(), CFE_ES_ValidateCDS(), CFE_ES_WaitForSystemState(), and CFE_ES_WriteToERLog().

13.23.2.2 CFE_ES_ResetDataPtr

```
CFE_ES_ResetData_t* CFE_ES_ResetDataPtr
```

Definition at line 73 of file cfe_es_start.c.

Referenced by CFE_ES_ClearERLogCmd(), CFE_ES_GetResetType(), CFE_ES_HousekeepingCmd(), CFE_ES↩ _ProcessCoreException(), CFE_ES_ResetCFE(), CFE_ES_ResetPRCountCmd(), CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetupPerfVariables(), CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogClear_Unsync(), CFE_ES↩ _SysLogReadData(), CFE_ES_SysLogReadStart_Unsync(), CFE_ES_SysLogSetMode(), CFE_ES_TaskInit(), and CFE_ES_WriteToERLog().

13.24 cfe/fsw/cfe-core/src/es/cfe_es_log.h File Reference

```
#include "cfe.h"
#include "cfe_es.h"
#include "cfe_es_global.h"
#include <stdarg.h>
```

Data Structures

- struct CFE_ES_SysLogReadBuffer_t

  Buffer structure for reading data out of the Syslog.

Macros

- #define CFE_ES_MAX_SYSLOG_MSG_SIZE (CFE_MISSION_EVS_MAX_MESSAGE_LENGTH + CFE_TIM↩ E_PRINTED_STRING_SIZE + 2)
- #define CFE_ES_SYSLOG_READ_BUFFER_SIZE (3 ∗ CFE_ES_MAX_SYSLOG_MSG_SIZE)
- #define CFE_ES_SYSLOG_APPEND(LogString)

  Self-synchronized macro to call CFE_ES_SysLogAppend_Unsync.

Functions

- void CFE_ES_SysLogClear_Unsync (void)

  Clear system log.
- void CFE_ES_SysLogReadStart_Unsync (CFE_ES_SysLogReadBuffer_t ∗Buffer)

  Begin reading the system log.
- int32 CFE_ES_SysLogWrite_Unsync (const char ∗SpecStringPtr,...)

  Write a printf-style formatted string to the system log.
- int32 CFE_ES_SysLogAppend_Unsync (const char ∗LogString)

  Append a complete pre-formatted string to the ES SysLog.
- void CFE_ES_SysLogReadData (CFE_ES_SysLogReadBuffer_t ∗Buffer)

  Read data from the system log buffer into the local buffer.
- int32 CFE_ES_SysLogSetMode (CFE_ES_LogMode_Enum_t Mode)

  Sets the operating mode of the system log buffer.
- void CFE_ES_SysLog_vsnprintf (char ∗Buffer, size_t BufferSize, const char ∗SpecStringPtr, va_list ArgPtr)

  Format a message intended for output to the system log.
- void CFE_ES_SysLog_snprintf (char ∗Buffer, size_t BufferSize, const char ∗SpecStringPtr,...) OS_PRINTF(3

  Format a message intended for output to the system log.
- void int32 CFE_ES_SysLogDump (const char ∗Filename)

  Write the contents of the syslog to a disk file.
- int32 CFE_ES_PerfLogClear (void)
- void CFE_ES_PerfLogDump (void)
- int32 CFE_ES_WriteToERLog (uint32 EntryType, uint32 ResetType, uint32 ResetSubtype, const char ∗Description, const uint32 ∗Context, uint32 ContextSize)
- int32 CFE_ES_ERLogDump (const char ∗Filename)

13.24.1 Macro Definition Documentation

13.24.1.1 CFE_ES_MAX_SYSLOG_MSG_SIZE

```
#define CFE_ES_MAX_SYSLOG_MSG_SIZE (CFE_MISSION_EVS_MAX_MESSAGE_LENGTH + CFE_TIME_PRINTED_STRING↩
_SIZE + 2)
```

Buffer size for system log messages

This is based on the EVS maximum event message size, plus a time stamp and required extra formatting characters.

Two extra characters are necessary:

- for the space between the timestamp and the message in the system log
- to enforce a newline character at the end of the string

note that a null terminator byte is accounted for in "CFE_TIME_PRINTED_STRING_SIZE"

Definition at line 66 of file cfe_es_log.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_DeleteCDS(), CFE_ES_PutPool↩Buf(), CFE_ES_SysLogWrite_Unsync(), and CFE_ES_WriteToSysLog().

13.24.1.2 CFE_ES_SYSLOG_APPEND

```
#define CFE_ES_SYSLOG_APPEND(
               LogString )
```

Value:

```
{                                                \
           CFE_ES_LockSharedData(__func__, __LINE__);      \
           CFE_ES_SysLogAppend_Unsync(LogString);          \
           CFE_ES_UnlockSharedData(__func__, __LINE__);    \
       }
```

Calls CFE_ES_SysLogAppend_Unsync() with appropriate synchronization. It will acquire the shared data lock and release it after appending the log.

This is implemented as a macro such that the "__func__" and "__LINE__" directives will reflect the actual place that the append was done, rather than where this wrapper was defined.

See also

CFE_ES_SysLogAppend_Unsync()

Definition at line 103 of file cfe_es_log.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_DeleteCDS(), and CFE_ES_Put↩PoolBuf().

13.24.1.3 CFE_ES_SYSLOG_READ_BUFFER_SIZE

```
#define CFE_ES_SYSLOG_READ_BUFFER_SIZE (3 * CFE_ES_MAX_SYSLOG_MSG_SIZE)
```

Size of the syslog "dump buffer"

This is a temporary buffer that serves as a holding place for syslog data as it is being dumped to a file on disk. Since disks are comparatively slow and access to the syslog buffer must be synchronized, copying to a temporary buffer first significantly decreases the amount of time that the syslog is locked after a file dump is requested.

This buffer also reflects the Syslog "burst size" that is guaranteed to be safe for concurrent writes and reads/dump operations. If applications Log more than this amount of data in less time than it takes to write this amount of data to disk, then some log messages may be corrupt or lost in the output file.

Note

If contention occurs where applications would overwrite logs that are still being "read" by a dump process, the realtime applications are given preference and therefore NOT blocked. Design preference is given to applications over the absolute integrity of the dump file.

Definition at line 88 of file cfe_es_log.h.

13.24.2 Function Documentation

13.24.2.1 CFE_ES_ERLogDump()

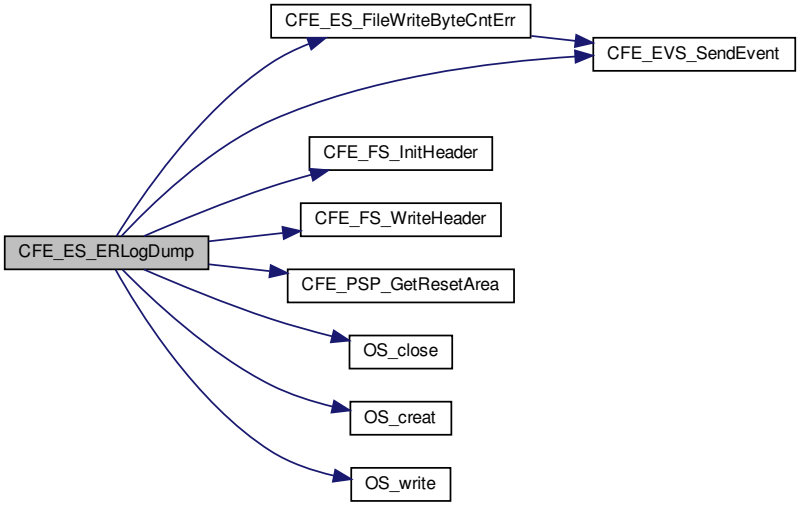
```
int32 CFE_ES_ERLogDump (
            const char * Filename )
```

Definition at line 1600 of file cfe_es_task.c.

References CFE_ES_ER_LOG_DESC, CFE_ES_ERLOG2_EID, CFE_ES_ERLOG2_ERR_EID, CFE_ES_FILE_IO_ERR, CFE_ES_FileWriteByteCntErr(), CFE_ES_RST_ACCESS_EID, CFE_ES_RST_ACCESS_ERR, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_ERLOG, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_ER_LOG_ENTRIES, CFE_PSP_GetResetArea(), CFE_PSP_SUCCESS, CFE_SUCCESS, OS_close(), OS_creat(), OS_write(), and OS_WRITE_ONLY.

Referenced by CFE_ES_WriteERLogCmd().

Here is the call graph for this function:



13.24.2.2 CFE_ES_PerfLogClear()

```
int32 CFE_ES_PerfLogClear (
            void  )
```

13.24.2.3 CFE_ES_PerfLogDump()

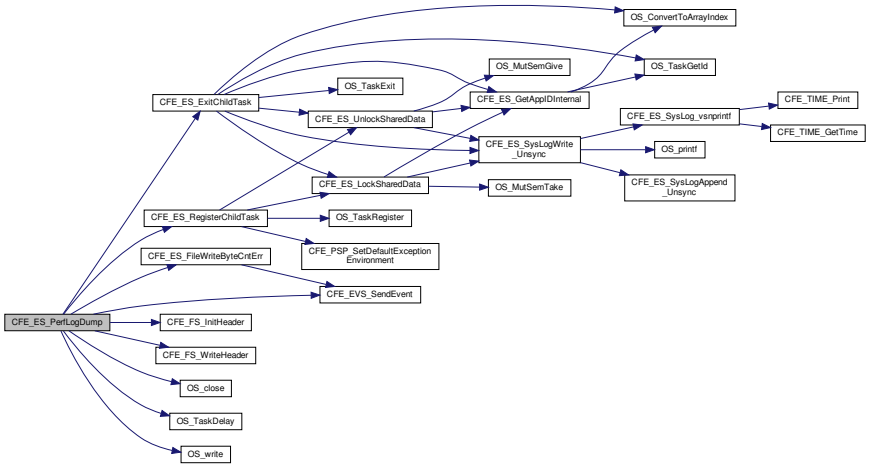
```
void CFE_ES_PerfLogDump (
            void  )
```

Definition at line 256 of file cfe_es_perf.c.

References CFE_ES_ExitChildTask(), CFE_ES_FileWriteByteCntErr(), CFE_ES_PERF_DATAWRITTEN_EID, CF↩E_ES_PERF_LOG_DESC, CFE_ES_RegisterChildTask(), CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_PERFDATA, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_PE↩RF_CHILD_MS_DELAY, CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS, CFE_ES_PerfData_t::DataBuffer, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfLogDump_t::DataFileDescriptor, CFE_ES_PerfLogDump_t::↩DataFileName, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfData_t::MetaData, OS_close(), OS_TaskDelay(), and OS_write().

Referenced by CFE_ES_StopPerfDataCmd().

Here is the call graph for this function:



13.24.2.4 CFE_ES_SysLog_snprintf()

```
void CFE_ES_SysLog_snprintf (
            char * Buffer,
            size_t BufferSize,
            const char * SpecStringPtr,
             ...  )
```

Identical to the CFE_ES_SysLog_vsnprintf() call but with a variable argument set, for use in functions that need to directly handle a log message string.

Similar in definition to the "snprintf()" C library call.

Parameters

Buffer	User supplied buffer to output formatted sting into
BufferSize	Size of "Buffer" parameter. Should be greater than (CFE_TIME_PRINTED_STRING_SIZE+2)
SpecStringPtr	Printf-style format string

See also

CFE_ES_SysLogAppend_Unsync()

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_DeleteCDS(), and CFE_ES_Put↩ PoolBuf().

13.24.2.5 CFE_ES_SysLog_vsnprintf()

```
void CFE_ES_SysLog_vsnprintf (
            char * Buffer,
            size_t BufferSize,
            const char * SpecStringPtr,
            va_list ArgPtr )
```

This function prepares a complete message for passing into CFE_ES_SysLogAppend_Unsync(), based on the given vsnprintf-style specification string and argument list.

The message is prefixed with a time stamp based on the current time, followed by the caller-specified string. An ending newline and terminating null character are both ensured on the output string.

To account for the timestamp, newline, and terminating null character, the supplied buffer must be greater than (CFE↩ _TIME_PRINTED_STRING_SIZE+2) to get a useful output. Any user-specified output string will be truncated to fit into the remaining space.

Parameters

Buffer	User supplied buffer to output formatted sting into
BufferSize	Size of "Buffer" parameter. Should be greater than (CFE_TIME_PRINTED_STRING_SIZE+2)
SpecStringPtr	Printf-style format string
ArgPtr	Variable argument list as obtained by va_start() in the caller

See also

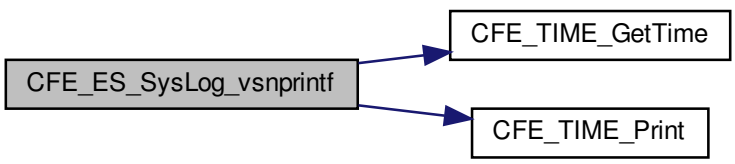
CFE_ES_SysLogAppend_Unsync()

Definition at line 376 of file cfe_es_syslog.c.

References CFE_TIME_GetTime(), CFE_TIME_Print(), and CFE_TIME_PRINTED_STRING_SIZE.

Referenced by CFE_ES_SysLog_snprintf(), CFE_ES_SysLogWrite_Unsync(), and CFE_ES_WriteToSysLog().

Here is the call graph for this function:



13.24.2.6 CFE_ES_SysLogAppend_Unsync()

```
int32 CFE_ES_SysLogAppend_Unsync (
            const char * LogString )
```

The new message will be copied to the current write location in the system log buffer. If there is not sufficient space to completely store the message, then the behavior depends on the "LogMode" setting.

If "LogMode" is set to DISCARD, then the message will be truncated to fit in the available space, or completely discarded if no space exists.

If "LogMode" is set to OVERWRITE, then the oldest message(s) in the system log will be overwritten with this new message.

Parameters

LogString	Message to append

Note

This function requires external thread synchronization

See also

CFE_ES_SysLogSetMode()

Definition at line 143 of file cfe_es_syslog.c.

References CFE_ES_ERR_SYS_LOG_FULL, CFE_ES_ERR_SYS_LOG_TRUNCATED, CFE_ES_LogMode_OVE↩RWRITE, CFE_ES_ResetDataPtr, CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, CFE_SUCCESS, CFE_TIME_PR↩INTED_STRING_SIZE, CFE_ES_ResetData_t::SystemLog, CFE_ES_ResetData_t::SystemLogEndIdx, CFE_ES_↩ResetData_t::SystemLogEntryNum, CFE_ES_ResetData_t::SystemLogMode, and CFE_ES_ResetData_t::System↩LogWriteIdx.

Referenced by CFE_ES_SysLogWrite_Unsync(), and CFE_ES_WriteToSysLog().

13.24.2.7 CFE_ES_SysLogClear_Unsync()

```
void CFE_ES_SysLogClear_Unsync (
            void  )
```

This discards the entire system log buffer and resets internal index values

Note

This function requires external thread synchronization

Definition at line 88 of file cfe_es_syslog.c.

References CFE_ES_ResetDataPtr, CFE_ES_ResetData_t::SystemLogEndIdx, CFE_ES_ResetData_t::SystemLog↩EntryNum, and CFE_ES_ResetData_t::SystemLogWriteIdx.

Referenced by CFE_ES_ClearSyslogCmd().

13.24.2.8 CFE_ES_SysLogDump()

```
void int32 CFE_ES_SysLogDump (
            const char * Filename )
```

Writes the current contents of the syslog buffer to a file specified by the Filename parameter. The log messages will be written to the file in the same order in which they were written into the syslog buffer.

A snapshot of the log indices is taken at the beginning of the writing process. Additional log entries added after this (e.g. from applications calling CFE_ES_WriteToSyslog() after starting a syslog dump) will not be included in the dump file.

Note that preference is given to the realtime application threads over any pending log read activities, such as a dumping to a file. The design of this function can tolerate a limited level of logging activity while the dump is in progress without any negative side effects. However, a significant "flood" of log messages may corrupt the output file, by overwriting older data before it has actually been written.

Parameters

Filename	Output file to write

Returns

CFE_SUCCESS if successful, or an appropriate error code from cfe_error.h

See also

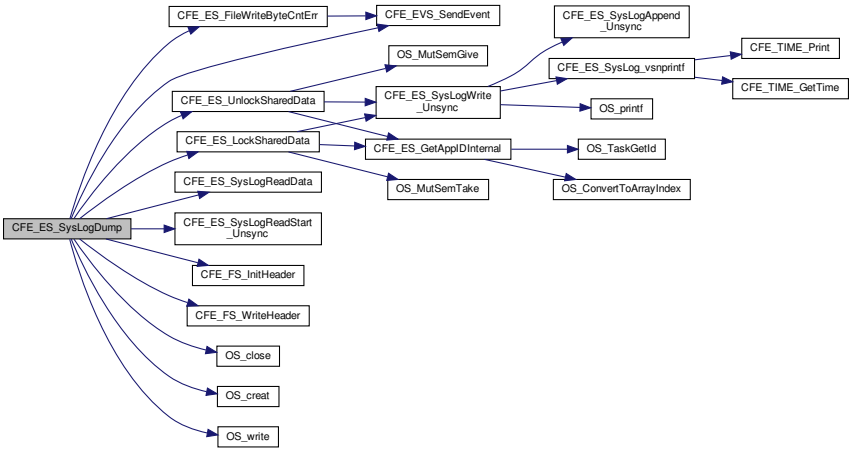
CFE_ES_SYSLOG_READ_BUFFER_SIZE

Definition at line 469 of file cfe_es_syslog.c.

References CFE_ES_FILE_IO_ERR, CFE_ES_FileWriteByteCntErr(), CFE_ES_LockSharedData(), CFE_ES_SYS↩
_LOG_DESC, CFE_ES_SYSLOG2_EID, CFE_ES_SYSLOG2_ERR_EID, CFE_ES_SysLogReadData(), CFE_ES_↩
SysLogReadStart_Unsync(), CFE_ES_TaskData, CFE_ES_UnlockSharedData(), CFE_EVS_EventType_DEBUG, C↩
FE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_SYSLOG, C↩
FE_FS_WriteHeader(), CFE_SUCCESS, CFE_ES_TaskData_t::HkPacket, OS_close(), OS_creat(), OS_write(), OS_↩
WRITE_ONLY, CFE_ES_HousekeepingTlm_t::Payload, and CFE_ES_HousekeepingTlm_Payload_t::SysLogEntries.

Referenced by CFE_ES_WriteSyslogCmd().

Here is the call graph for this function:



13.24.2.9 CFE_ES_SysLogReadData()

```
void CFE_ES_SysLogReadData (
            CFE_ES_SysLogReadBuffer_t * Buffer )
```

Prior to calling this function, the buffer structure should be initialized using CFE_ES_SysLogReadStart_Unsync()

This copies the data from the syslog memory space into the local buffer, starting from the end of the previously read data. To read the complete system log, this function should be called repeatedly until the "BlockSize" member in the returned buffer is returned as zero, indicating there is no more data in the syslog.

There is no specific external synchronization requirement on this function, since copies of the relevant log indices are kept in the buffer structure itself. However, if system log data is overwritten between calls to this function, it may result in undefined data being returned to the caller.

Therefore, in cases where it is critically important to read log message data, the lock should be held for the entire procedure (initialization through complete read). However this may have significant realtime implications, so it is not the required mode of operation.

Parameters

Buffer	A local buffer which will be filled with data from the log buffer

Definition at line 302 of file cfe_es_syslog.c.

References CFE_ES_SysLogReadBuffer_t::BlockSize, CFE_ES_ResetDataPtr, CFE_ES_SysLogReadBuffer_t::Data, CFE_ES_SysLogReadBuffer_t::EndIdx, CFE_ES_SysLogReadBuffer_t::LastOffset, CFE_ES_SysLogReadBuffer_t::↩ SizeLeft, and CFE_ES_ResetData_t::SystemLog.

Referenced by CFE_ES_SysLogDump().

13.24.2.10 CFE_ES_SysLogReadStart_Unsync()

```
void CFE_ES_SysLogReadStart_Unsync (
            CFE_ES_SysLogReadBuffer_t * Buffer )
```

This a helper function is intended to assist with the "Write" command to dump the contents of the syslog to a disk file. This locates the oldest complete log message currently contained in the buffer.

The oldest log message may be overwritten when any application calls CFE_ES_WriteToSysLog() if set to OVERWRITE mode.

This function only locates the first message, it does not actually copy any data to the supplied buffer. The CFE_ES_SysLogReadData() should be called to read log data.

Parameters

Buffer	A local buffer which will be initialized to the start of the log buffer

Note

This function requires external thread synchronization

See also

CFE_ES_SysLogReadData()

Definition at line 107 of file cfe_es_syslog.c.

References CFE_ES_SysLogReadBuffer_t::BlockSize, CFE_ES_ResetDataPtr, CFE_ES_SysLogReadBuffer_t::EndIdx, CFE_ES_SysLogReadBuffer_t::LastOffset, CFE_ES_SysLogReadBuffer_t::SizeLeft, CFE_ES_ResetData_t::SystemLog, CFE_ES_ResetData_t::SystemLogEndIdx, and CFE_ES_ResetData_t::SystemLogWriteIdx.

Referenced by CFE_ES_SysLogDump().

13.24.2.11 CFE_ES_SysLogSetMode()

```
int32 CFE_ES_SysLogSetMode (
            CFE_ES_LogMode_Enum_t Mode )
```

The operating mode of the system log controls its behavior once filled to the point where additional messages can no longer be stored.

If "Mode" is set to DISCARD, then the message will be truncated to fit in the available space, or completely discarded if no space exists.

If "Mode" is set to OVERWRITE, then the oldest message(s) in the system log will be overwritten with this new message.

Note

Switching from OVERWRITE to DISCARD mode may take effect immediately, as the setting only takes effect when the buffer "wrap-point" is reached at the end.

Parameters

Mode	The desired operating mode

Returns

CFE_SUCCESS if set successfully

Definition at line 352 of file cfe_es_syslog.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_LogMode_DISCARD, CFE_ES_LogMode_OVERWRITE, CFE_↩ ES_ResetDataPtr, CFE_SUCCESS, and CFE_ES_ResetData_t::SystemLogMode.

Referenced by CFE_ES_OverWriteSyslogCmd().

13.24.2.12 CFE_ES_SysLogWrite_Unsync()

```
int32 CFE_ES_SysLogWrite_Unsync (
            const char * SpecStringPtr,
             ...  )
```

This is a drop-in replacement for the existing CFE_ES_WriteToSysLog() API that does *not* perform any synchronization or locking. It is intended for logging from within the ES subsystem where the appropriate lock is already held for other reasons.

Note

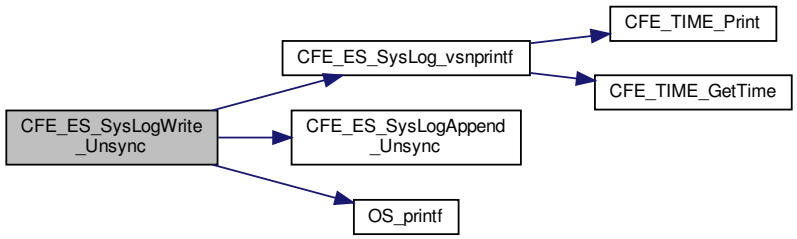
This function requires external thread synchronization

Definition at line 267 of file cfe_es_syslog.c.

References CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_SysLog_vsnprintf(), CFE_ES_SysLogAppend_Unsync(), and OS_printf().

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CleanupObjectCallback(), CFE_ES_↩ CreateCDSPool(), CFE_ES_CreateChildTask(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_Exit↩ App(), CFE_ES_ExitChildTask(), CFE_ES_GetTaskInfo(), CFE_ES_LockSharedData(), CFE_ES_Main(), CFE_ES↩ _RebuildCDSPool(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_SetupReset↩ Variables(), and CFE_ES_UnlockSharedData().

Here is the call graph for this function:



13.24.2.13 CFE_ES_WriteToERLog()

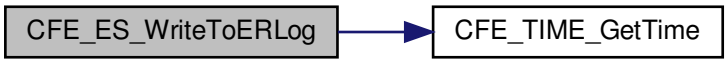
```
int32 CFE_ES_WriteToERLog (
            uint32 EntryType,
            uint32 ResetType,
            uint32 ResetSubtype,
            const char * Description,
            const uint32 * Context,
            uint32 ContextSize )
```

Definition at line 68 of file cfe_es_erlog.c.

References CFE_ES_ResetVariables_t::BootSource, CFE_ES_ERLog_t::BootSource, CFE_ES_Global, CFE_ES_ResetDataPtr, CFE_PLATFORM_ES_ER_LOG_ENTRIES, CFE_PLATFORM_ES_ER_LOG_MAX_CONTEXT_SIZE, CFE_SUCCESS, CFE_TIME_GetTime(), CFE_ES_ERLog_t::Context, CFE_ES_ERLog_t::ContextSize, CFE_ES_ERLog_t::DebugVars, CFE_ES_Global_t::DebugVars, CFE_ES_ERLog_t::Description, CFE_ES_ResetData_t::ERLog, CFE_ES_ResetData_t::ERLogEntries, CFE_ES_ResetData_t::ERLogIndex, CFE_ES_ERLog_t::LogEntryType, CFE_ES_ResetVariables_t::MaxProcessorResetCount, CFE_ES_ERLog_t::MaxProcessorResetCount, NULL, CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_ERLog_t::ProcessorResetCount, CFE_ES_ERLog_t::ResetSubtype, CFE_ES_ERLog_t::ResetType, CFE_ES_ResetData_t::ResetVars, and CFE_ES_ERLog_t::TimeCode.

Referenced by CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), and CFE_ES_SetupResetVariables().

Here is the call graph for this function:



13.25 cfe/fsw/cfe-core/src/es/cfe_es_objtab.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es_global.h"
#include "cfe_es_start.h"
```

Variables

- CFE_ES_ObjectTable_t CFE_ES_ObjectTable [CFE_PLATFORM_ES_OBJECT_TABLE_SIZE]

13.25.1 Variable Documentation

13.25.1.1 CFE_ES_ObjectTable

```
CFE_ES_ObjectTable_t CFE_ES_ObjectTable[CFE_PLATFORM_ES_OBJECT_TABLE_SIZE]
```

Definition at line 49 of file cfe_es_objtab.c.

Referenced by CFE_ES_CreateObjects().

13.26 cfe/fsw/cfe-core/src/es/cfe_es_perf.c File Reference

```
#include "osapi.h"
#include "private/cfe_private.h"
#include "cfe_es_perf.h"
#include "cfe_es_log.h"
#include "cfe_es_global.h"
#include "cfe_es_start.h"
#include "cfe_es_events.h"
#include "cfe_es_task.h"
#include "cfe_fs.h"
#include "cfe_psp.h"
#include <string.h>
```

Functions

- void CFE_ES_SetupPerfVariables (uint32 ResetType)
- int32 CFE_ES_StartPerfDataCmd (const CFE_ES_StartPerfData_t *data)
- int32 CFE_ES_StopPerfDataCmd (const CFE_ES_StopPerfData_t *data)
- void CFE_ES_PerfLogDump (void)
- int32 CFE_ES_SetPerfFilterMaskCmd (const CFE_ES_SetPerfFilterMask_t *data)
- int32 CFE_ES_SetPerfTriggerMaskCmd (const CFE_ES_SetPerfTriggerMask_t *data)
- void CFE_ES_PerfLogAdd (uint32 Marker, uint32 EntryExit)

  Function called by CFE_ES_PerfLogEntry and CFE_ES_PerfLogExit macros.

Variables

- CFE_ES_PerfData_t * Perf
- CFE_ES_PerfLogDump_t CFE_ES_PerfLogDumpStatus

13.26.1 Function Documentation

13.26.1.1 CFE_ES_PerfLogAdd()

```
void CFE_ES_PerfLogAdd (
            uint32 Marker,
            uint32 EntryExit )
```

Description

This function logs the entry and exit marker for the specified `id`. This function is used by the Software Performance Analysis tool (see section 5.15).

Assumptions, External Events, and Notes:

None

Parameters

in	*Marker*	Identifier of the specific event or marker.
in	*EntryExit*	Used to specify Entry(0) or Exit(1)

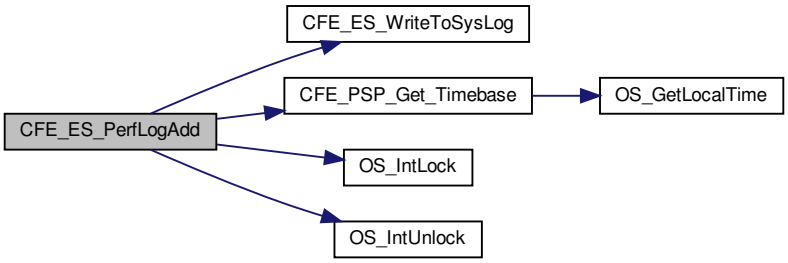
See also

CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit

Definition at line 415 of file cfe_es_perf.c.

References CFE_ES_PERF_IDLE, CFE_ES_PERF_TRIGGER_CENTER, CFE_ES_PERF_TRIGGER_END, CFE_ES_PERF_TRIGGER_START, CFE_ES_PERF_TRIGGERED, CFE_ES_PERF_WAITING_FOR_TRIGGER, CFE_ES_TEST_LONG_MASK, CFE_ES_WriteToSysLog(), CFE_MISSION_ES_PERF_EXIT_BIT, CFE_MISSION_ES_PERF_MAX_IDS, CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE, CFE_PSP_Get_Timebase(), CFE_ES_PerfDataEntry_t::Data, CFE_ES_PerfData_t::DataBuffer, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfMetaData_t::FilterMask, CFE_ES_PerfMetaData_t::InvalidMarkerReported, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, OS_IntLock(), OS_IntUnlock(), CFE_ES_PerfMetaData_t::State, CFE_ES_PerfDataEntry_t::TimerLower32, CFE_ES_PerfDataEntry_t::TimerUpper32, CFE_ES_PerfMetaData_t::TriggerCount, and CFE_ES_PerfMetaData_t::TriggerMask.

Here is the call graph for this function:



13.26.1.2 CFE_ES_PerfLogDump()

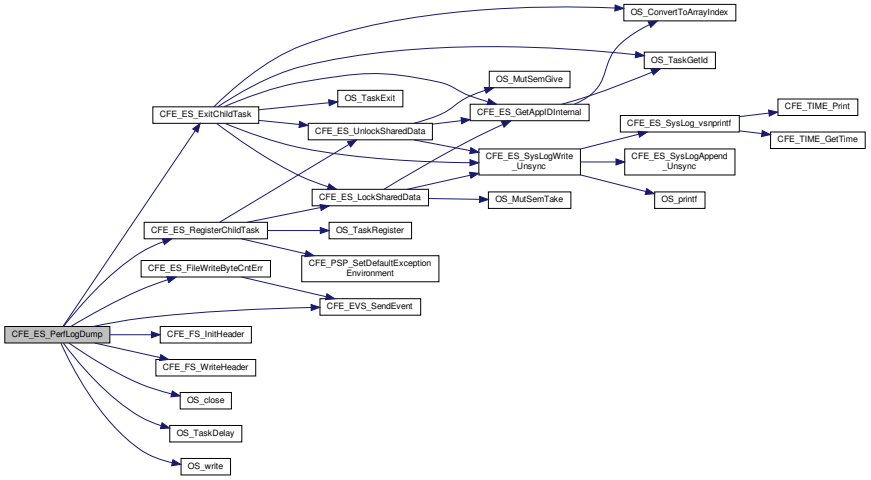
```
void CFE_ES_PerfLogDump (
            void  )
```

Definition at line 256 of file cfe_es_perf.c.

References CFE_ES_ExitChildTask(), CFE_ES_FileWriteByteCntErr(), CFE_ES_PERF_DATAWRITTEN_EID, CF↩E_ES_PERF_LOG_DESC, CFE_ES_RegisterChildTask(), CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_PERFDATA, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_PE↩RF_CHILD_MS_DELAY, CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS, CFE_ES_PerfData_t::DataBuffer, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfLogDump_t::DataFileDescriptor, CFE_ES_PerfLogDump_t::↩DataFileName, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfData_t::MetaData, OS_close(), OS_TaskDelay(), and OS_write().

Referenced by CFE_ES_StopPerfDataCmd().

Here is the call graph for this function:



13.26.1.3 CFE_ES_SetPerfFilterMaskCmd()

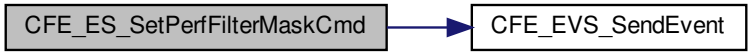
```
int32 CFE_ES_SetPerfFilterMaskCmd (
            const CFE_ES_SetPerfFilterMask_t * data )
```

Definition at line 336 of file cfe_es_perf.c.

References CFE_ES_PERF_32BIT_WORDS_IN_MASK, CFE_ES_PERF_FILTMSKCMD_EID, CFE_ES_PERF_F↩ILTMSKERR_EID, CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EV↩S_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandError↩Counter, CFE_ES_PerfMetaData_t::FilterMask, CFE_ES_SetPerfFilterMaskCmd_Payload_t::FilterMask, CFE_ES↩_SetPerfFilterMaskCmd_Payload_t::FilterMaskNum, CFE_ES_PerfData_t::MetaData, and CFE_ES_SetPerfFilter↩Mask_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.26.1.4 CFE_ES_SetPerfTriggerMaskCmd()

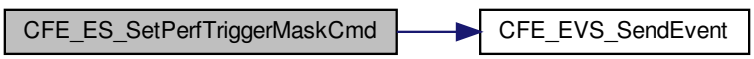
```
int32 CFE_ES_SetPerfTriggerMaskCmd (
            const CFE_ES_SetPerfTriggerMask_t * data )
```

Definition at line 366 of file cfe_es_perf.c.

References CFE_ES_PERF_32BIT_WORDS_IN_MASK, CFE_ES_PERF_TRIGMSKCMD_EID, CFE_ES_PERF_TRIGMSKERR_EID, CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_PerfData_t::MetaData, CFE_ES_SetPerfTriggerMask_t::Payload, CFE_ES_PerfMetaData_t::TriggerMask, CFE_ES_SetPerfTrigMaskCmd_Payload_t::TriggerMask, and CFE_ES_SetPerfTrigMaskCmd_Payload_t::TriggerMaskNum.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.26.1.5 CFE_ES_SetupPerfVariables()

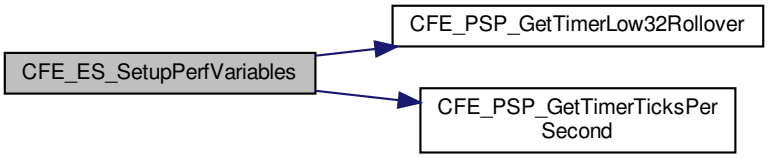
```
void CFE_ES_SetupPerfVariables (
            uint32 ResetType )
```

Definition at line 62 of file cfe_es_perf.c.

References CFE_ES_PERF_32BIT_WORDS_IN_MASK, CFE_ES_PERF_IDLE, CFE_ES_PERF_TRIGGER_START, CFE_ES_ResetDataPtr, CFE_PLATFORM_ES_PERF_FILTMASK_INIT, CFE_PLATFORM_ES_PERF_TRIGMASK_INIT, CFE_PSP_GetTimerLow32Rollover(), CFE_PSP_GetTimerTicksPerSecond(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_ES_PerfLogDump_t::ChildID, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfLogDump_t::DataFileName, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfMetaData_t::Endian, CFE_ES_PerfMetaData_t::FilterMask, CFE_ES_PerfMetaData_t::FilterTriggerMaskSize, CFE_ES_PerfMetaData_t::InvalidMarkerReported, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, CFE_ES_ResetData_t::Perf, CFE_ES_PerfMetaData_t::State, CFE_ES_PerfMetaData_t::TimerLow32Rollover, CFE_ES_PerfMetaData_t::TimerTicksPerSecond, CFE_ES_PerfMetaData_t::TriggerCount, CFE_ES_PerfMetaData_t::TriggerMask, and CFE_ES_PerfMetaData_t::Version.

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.26.1.6 CFE_ES_StartPerfDataCmd()

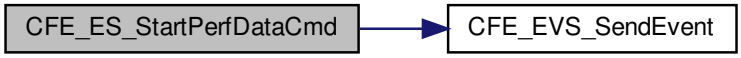
```
int32 CFE_ES_StartPerfDataCmd (
            const CFE_ES_StartPerfData_t * data )
```

Definition at line 126 of file cfe_es_perf.c.

References CFE_ES_PERF_MAX_MODES, CFE_ES_PERF_STARTCMD_EID, CFE_ES_PERF_STARTCMD_ERR_EID, CFE_ES_PERF_STARTCMD_TRIG_ERR_EID, CFE_ES_PERF_TRIGGER_END, CFE_ES_PERF_TRIGGER_START, CFE_ES_PERF_WAITING_FOR_TRIGGER, CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfMetaData_t::InvalidMarkerReported, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, CFE_ES_StartPerfData_t::Payload, CFE_ES_PerfMetaData_t::State, CFE_ES_PerfMetaData_t::TriggerCount, and CFE_ES_StartPerfCmd_Payload_t::TriggerMode.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.26.1.7 CFE_ES_StopPerfDataCmd()

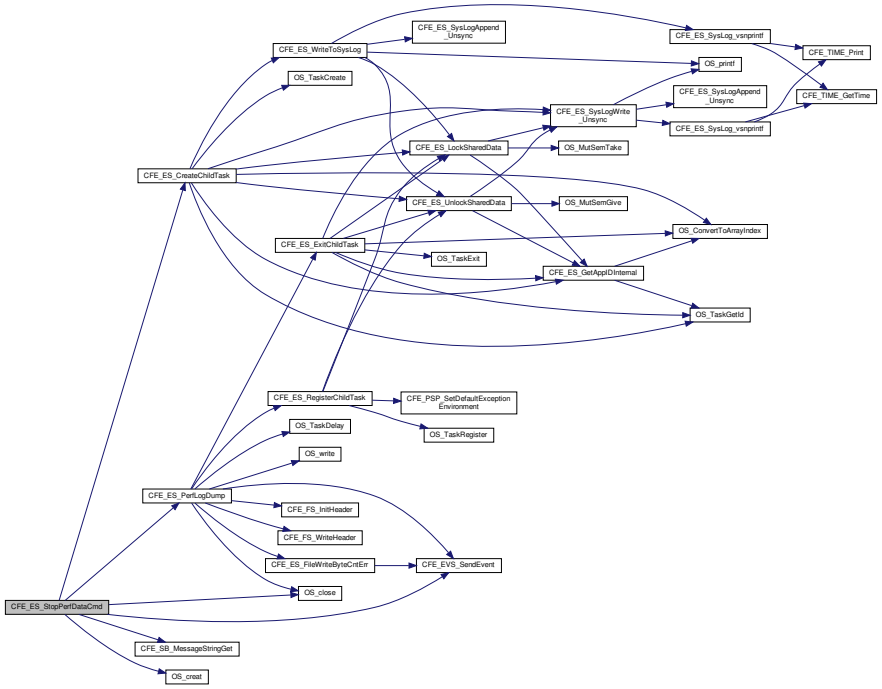
```
int32 CFE_ES_StopPerfDataCmd (
            const CFE_ES_StopPerfData_t * data )
```

Definition at line 176 of file cfe_es_perf.c.

References CFE_ES_CreateChildTask(), CFE_ES_PERF_CHILD_FLAGS, CFE_ES_PERF_CHILD_NAME, CFE_ES_PERF_CHILD_STACK_PTR, CFE_ES_PERF_IDLE, CFE_ES_PERF_LOG_ERR_EID, CFE_ES_PERF_STOPCMD_EID, CFE_ES_PERF_STOPCMD_ERR1_EID, CFE_ES_PERF_STOPCMD_ERR2_EID, CFE_ES_PerfLogDump(), CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME, CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY, CFE_PLATFORM_ES_PERF_CHILD_PRIORITY, CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE, CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_PerfLogDump_t::ChildID, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfLogDump_t::DataFileDescriptor, CFE_ES_PerfLogDump_t::DataFileName, CFE_ES_StopPerfCmd_Payload_t::DataFileName, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfData_t::MetaData, OS_close(), OS_creat(), OS_MAX_PATH_LEN, OS_WRITE_ONLY, CFE_ES_StopPerfData_t::Payload, and CFE_ES_PerfMetaData_t::State.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.26.2 Variable Documentation

13.26.2.1 CFE_ES_PerfLogDumpStatus

```
CFE_ES_PerfLogDump_t CFE_ES_PerfLogDumpStatus
```

Definition at line 50 of file cfe_es_perf.c.

Referenced by CFE_ES_HousekeepingCmd().

13.26.2.2 Perf

```
CFE_ES_PerfData_t* Perf
```

Definition at line 49 of file cfe_es_perf.c.

13.27 cfe/fsw/cfe-core/src/es/cfe_es_perf.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_es.h"
#include "cfe_es_msg.h"
#include "cfe_es_events.h"
#include "cfe_sb.h"
#include "cfe_evs.h"
#include "cfe_perfids.h"
#include "cfe_psp.h"
```

Data Structures

- struct CFE_ES_PerfLogDump_t

Macros

- #define CFE_ES_PERF_CHILD_NAME "ES_PerfFileWriter"
- #define CFE_ES_PERF_CHILD_STACK_PTR 0
- #define CFE_ES_PERF_CHILD_FLAGS 0

Enumerations

- enum CFE_ES_PerfState_t { CFE_ES_PERF_IDLE = 0, CFE_ES_PERF_WAITING_FOR_TRIGGER, CFE_ES_PERF_TRIGGERED, CFE_ES_PERF_MAX_STATES }
- enum CFE_ES_PerfMode_t { CFE_ES_PERF_TRIGGER_START = 0, CFE_ES_PERF_TRIGGER_CENTER, CFE_ES_PERF_TRIGGER_END, CFE_ES_PERF_MAX_MODES }

Variables

- CFE_ES_PerfLogDump_t CFE_ES_PerfLogDumpStatus

13.27.1 Macro Definition Documentation

13.27.1.1 CFE_ES_PERF_CHILD_FLAGS

```
#define CFE_ES_PERF_CHILD_FLAGS 0
```

Definition at line 54 of file cfe_es_perf.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.27.1.2 CFE_ES_PERF_CHILD_NAME

```
#define CFE_ES_PERF_CHILD_NAME "ES_PerfFileWriter"
```

Definition at line 52 of file cfe_es_perf.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.27.1.3 CFE_ES_PERF_CHILD_STACK_PTR

```
#define CFE_ES_PERF_CHILD_STACK_PTR 0
```

Definition at line 53 of file cfe_es_perf.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.27.2 Enumeration Type Documentation

13.27.2.1 CFE_ES_PerfMode_t

```
enum CFE_ES_PerfMode_t
```

Enumerator

CFE_ES_PERF_TRIGGER_START	
CFE_ES_PERF_TRIGGER_CENTER	
CFE_ES_PERF_TRIGGER_END	
CFE_ES_PERF_MAX_MODES	

Definition at line 64 of file cfe_es_perf.h.

13.27.2.2 CFE_ES_PerfState_t

```
enum CFE_ES_PerfState_t
```

Enumerator

CFE_ES_PERF_IDLE	
CFE_ES_PERF_WAITING_FOR_TRIGGER	
CFE_ES_PERF_TRIGGERED	
CFE_ES_PERF_MAX_STATES	

Definition at line 57 of file cfe_es_perf.h.

13.27.3 Variable Documentation

13.27.3.1 CFE_ES_PerfLogDumpStatus

```
CFE_ES_PerfLogDump_t CFE_ES_PerfLogDumpStatus
```

Definition at line 50 of file cfe_es_perf.c.

Referenced by CFE_ES_HousekeepingCmd().

13.28 cfe/fsw/cfe-core/src/es/cfe_es_shell.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es_global.h"
#include "cfe_es_apps.h"
#include "cfe_es_shell.h"
#include "cfe_es_task.h"
#include "cfe_es_log.h"
#include "cfe_psp.h"
#include <string.h>
```

Macros

- #define CFE_ES_CHECKSIZE 3

Functions

- int32 CFE_ES_ShellOutputCommand (const char *CmdString, const char *Filename)
- int32 CFE_ES_ListApplications (int32 fd)
- int32 CFE_ES_ListTasks (int32 fd)
- static void CFE_ES_ShellCountObjectCallback (uint32 object_id, void *arg)
- int32 CFE_ES_ListResources (int32 fd)

13.28.1 Macro Definition Documentation

13.28.1.1 CFE_ES_CHECKSIZE

```
#define CFE_ES_CHECKSIZE 3
```

Definition at line 47 of file cfe_es_shell.c.

Referenced by CFE_ES_ShellOutputCommand().

13.28.2 Function Documentation

13.28.2.1 CFE_ES_ListApplications()

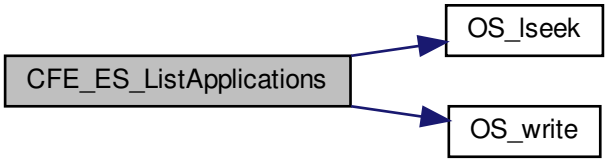
```
int32 CFE_ES_ListApplications (
            int32 fd )
```

Definition at line 238 of file cfe_es_shell.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_Global, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppStartParams_t::Name, OS_lseek(), OS_MAX_API_NAME, OS_SEEK_SET, OS_write(), and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_ShellOutputCommand().

Here is the call graph for this function:



13.28.2.2 CFE_ES_ListResources()

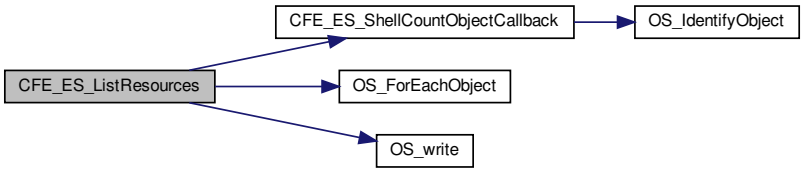
```
int32 CFE_ES_ListResources (
            int32 fd )
```

Definition at line 357 of file cfe_es_shell.c.

References CFE_ES_ShellCountObjectCallback(), CFE_SUCCESS, OS_ForEachObject(), OS_OBJECT_TYPE_OS↩
_BINSEM, OS_OBJECT_TYPE_OS_COUNTSEM, OS_OBJECT_TYPE_OS_MUTEX, OS_OBJECT_TYPE_OS_Q↩
UEUE, OS_OBJECT_TYPE_OS_STREAM, OS_OBJECT_TYPE_OS_TASK, OS_OBJECT_TYPE_USER, and OS_↩
write().

Referenced by CFE_ES_ShellOutputCommand().

Here is the call graph for this function:



13.28.2.3 CFE_ES_ListTasks()

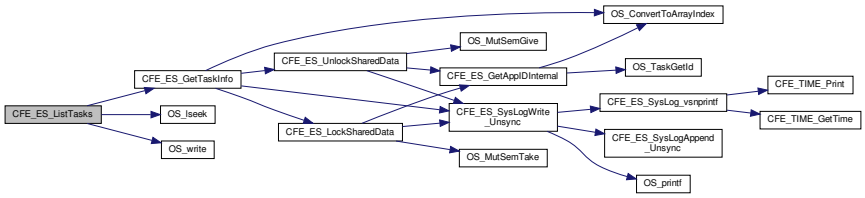
```
int32 CFE_ES_ListTasks (
            int32 fd )
```

Definition at line 273 of file cfe_es_shell.c.

References CFE_ES_TaskInfo_t::AppId, CFE_ES_TaskInfo_t::AppName, CFE_ES_GetTaskInfo(), CFE_ES_Global, CFE_SUCCESS, OS_lseek(), OS_MAX_API_NAME, OS_MAX_TASKS, OS_SEEK_SET, OS_write(), CFE_ES_↩
TaskRecord_t::RecordUsed, CFE_ES_TaskRecord_t::TaskId, CFE_ES_TaskInfo_t::TaskId, CFE_ES_TaskInfo_t::↩
TaskName, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_ShellOutputCommand().

Here is the call graph for this function:



13.28.2.4 CFE_ES_ShellCountObjectCallback()

```
static void CFE_ES_ShellCountObjectCallback (
             uint32 object_id,
             void * arg )  [static]
```

Definition at line 338 of file cfe_es_shell.c.

References OS_IdentifyObject(), and OS_OBJECT_TYPE_USER.

Referenced by CFE_ES_ListResources().

Here is the call graph for this function:

13.28.2.5 CFE_ES_ShellOutputCommand()

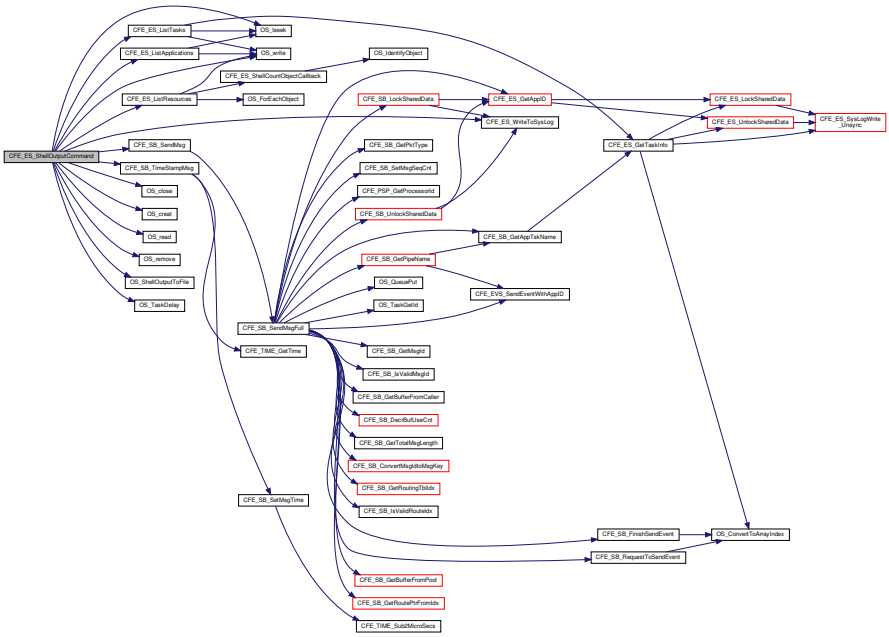
```
int32 CFE_ES_ShellOutputCommand (
             const char * CmdString,
             const char * Filename )
```

Definition at line 52 of file cfe_es_shell.c.

References CFE_ES_CHECKSIZE, CFE_ES_ERR_SHELL_CMD, CFE_ES_LIST_APPS_CMD, CFE_ES_LIST_RE↩ SOURCES_CMD, CFE_ES_LIST_TASKS_CMD, CFE_ES_ListApplications(), CFE_ES_ListResources(), CFE_ES_↩ ListTasks(), CFE_ES_TaskData, CFE_ES_WriteToSysLog(), CFE_MISSION_ES_MAX_SHELL_PKT, CFE_PLATF↩ ORM_ES_DEFAULT_SHELL_FILENAME, CFE_PLATFORM_ES_SHELL_OS_DELAY_MILLISEC, CFE_SB_Send↩ Msg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, NULL, OS_close(), OS_creat(), OS_FS_ERROR, OS_FS_S↩ UCCESS, OS_lseek(), OS_read(), OS_READ_WRITE, OS_remove(), OS_SEEK_END, OS_SEEK_SET, OS_Shell↩ OutputToFile(), OS_SUCCESS, OS_TaskDelay(), OS_write(), CFE_ES_ShellTlm_t::Payload, CFE_ES_ShellPacket_↩ Payload_t::ShellOutput, and CFE_ES_TaskData_t::ShellPacket.

Referenced by CFE_ES_ShellCmd().

Here is the call graph for this function:



13.29 cfe/fsw/cfe-core/src/es/cfe_es_shell.h File Reference

```
#include "cfe.h"
```

Macros

- #define CFE_ES_LIST_APPS_CMD "ES_ListApps"
- #define CFE_ES_LIST_RESOURCES_CMD "ES_ListResources"
- #define CFE_ES_LIST_TASKS_CMD "ES_ListTasks"

Functions

- int32 CFE_ES_ShellOutputCommand (const char ∗CmdString, const char ∗Filename)
- int32 CFE_ES_ListApplications (int32 fd)
- int32 CFE_ES_ListTasks (int32 fd)
- int32 CFE_ES_ListResources (int32 fd)

13.29.1 Macro Definition Documentation

13.29.1.1 CFE_ES_LIST_APPS_CMD

```
#define CFE_ES_LIST_APPS_CMD "ES_ListApps"
```

Definition at line 47 of file cfe_es_shell.h.

Referenced by CFE_ES_ShellOutputCommand().

13.29.1.2 CFE_ES_LIST_RESOURCES_CMD

```
#define CFE_ES_LIST_RESOURCES_CMD "ES_ListResources"
```

Definition at line 48 of file cfe_es_shell.h.

Referenced by CFE_ES_ShellOutputCommand().

13.29.1.3 CFE_ES_LIST_TASKS_CMD

```
#define CFE_ES_LIST_TASKS_CMD "ES_ListTasks"
```

Definition at line 49 of file cfe_es_shell.h.

Referenced by CFE_ES_ShellOutputCommand().

13.29.2 Function Documentation

13.29.2.1 CFE_ES_ListApplications()

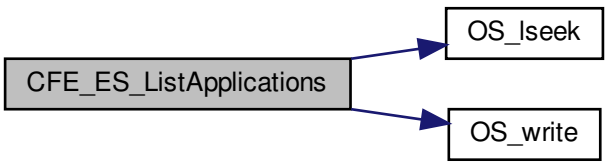
```
int32 CFE_ES_ListApplications (
            int32 fd )
```

Definition at line 238 of file cfe_es_shell.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_Global, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppStartParams_t::Name, OS_lseek(), OS_MAX_API_NAME, OS_SEEK_SET, OS_write(), and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_ShellOutputCommand().

Here is the call graph for this function:



13.29.2.2 CFE_ES_ListResources()

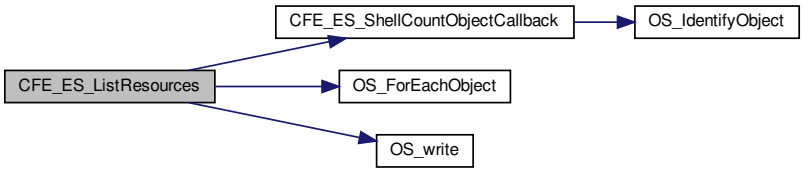
```
int32 CFE_ES_ListResources (
            int32 fd )
```

Definition at line 357 of file cfe_es_shell.c.

References CFE_ES_ShellCountObjectCallback(), CFE_SUCCESS, OS_ForEachObject(), OS_OBJECT_TYPE_OS↩
_BINSEM, OS_OBJECT_TYPE_OS_COUNTSEM, OS_OBJECT_TYPE_OS_MUTEX, OS_OBJECT_TYPE_OS_Q↩
UEUE, OS_OBJECT_TYPE_OS_STREAM, OS_OBJECT_TYPE_OS_TASK, OS_OBJECT_TYPE_USER, and OS_↩
write().

Referenced by CFE_ES_ShellOutputCommand().

Here is the call graph for this function:



13.29.2.3 CFE_ES_ListTasks()

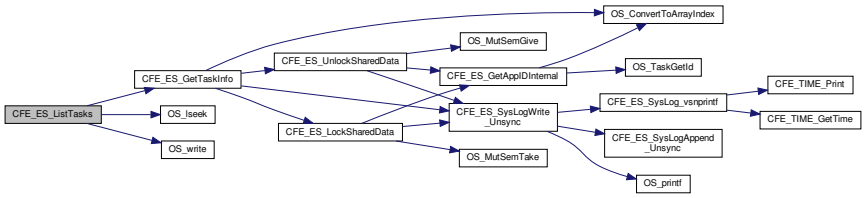
```
int32 CFE_ES_ListTasks (
            int32 fd )
```

Definition at line 273 of file cfe_es_shell.c.

References CFE_ES_TaskInfo_t::AppId, CFE_ES_TaskInfo_t::AppName, CFE_ES_GetTaskInfo(), CFE_ES_Global, CFE_SUCCESS, OS_lseek(), OS_MAX_API_NAME, OS_MAX_TASKS, OS_SEEK_SET, OS_write(), CFE_ES_↩
TaskRecord_t::RecordUsed, CFE_ES_TaskRecord_t::TaskId, CFE_ES_TaskInfo_t::TaskId, CFE_ES_TaskInfo_t::↩
TaskName, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_ShellOutputCommand().

Here is the call graph for this function:



13.29.2.4 CFE_ES_ShellOutputCommand()

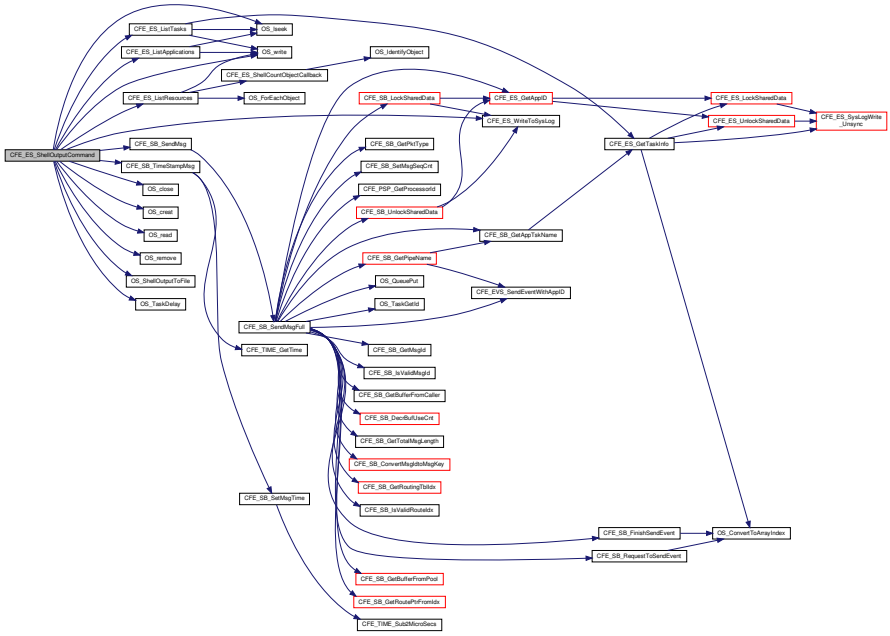
```
int32 CFE_ES_ShellOutputCommand (
            const char * CmdString,
            const char * Filename )
```

Definition at line 52 of file cfe_es_shell.c.

References CFE_ES_CHECKSIZE, CFE_ES_ERR_SHELL_CMD, CFE_ES_LIST_APPS_CMD, CFE_ES_LIST_RESOURCES_CMD, CFE_ES_LIST_TASKS_CMD, CFE_ES_ListApplications(), CFE_ES_ListResources(), CFE_ES_ListTasks(), CFE_ES_TaskData, CFE_ES_WriteToSysLog(), CFE_MISSION_ES_MAX_SHELL_PKT, CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME, CFE_PLATFORM_ES_SHELL_OS_DELAY_MILLISEC, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, NULL, OS_close(), OS_creat(), OS_FS_ERROR, OS_FS_SUCCESS, OS_lseek(), OS_read(), OS_READ_WRITE, OS_remove(), OS_SEEK_END, OS_SEEK_SET, OS_ShellOutputToFile(), OS_SUCCESS, OS_TaskDelay(), OS_write(), CFE_ES_ShellTlm_t::Payload, CFE_ES_ShellPacket_Payload_t::ShellOutput, and CFE_ES_TaskData_t::ShellPacket.

Referenced by CFE_ES_ShellCmd().

Here is the call graph for this function:



13.30 cfe/fsw/cfe-core/src/es/cfe_es_start.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_es.h"
#include "cfe_es_global.h"
#include "cfe_es_start.h"
#include "cfe_es_apps.h"
#include "cfe_es_log.h"
#include "cfe_psp.h"
#include <stdio.h>
#include <string.h>
```

Macros

- #define CFE_ES_PANIC_DELAY 500

Functions

- static int32 CFE_ES_MainTaskSyncDelay (uint32 AppStateId, uint32 TimeOutMilliseconds)
- void CFE_ES_Main (uint32 StartType, uint32 StartSubtype, uint32 ModeId, const char *StartFilePath)

  cFE Main Entry Point used by Board Support Package to start cFE
- void CFE_ES_SetupResetVariables (uint32 StartType, uint32 StartSubtype, uint32 BootSource)
- void CFE_ES_InitializeFileSystems (uint32 StartType)
- void CFE_ES_CreateObjects (void)

Variables

- CFE_ES_Global_t CFE_ES_Global
- CFE_ES_ResetData_t * CFE_ES_ResetDataPtr

13.30.1 Macro Definition Documentation

13.30.1.1 CFE_ES_PANIC_DELAY

```
#define CFE_ES_PANIC_DELAY 500
```

Definition at line 62 of file cfe_es_start.c.

Referenced by CFE_ES_CreateObjects(), CFE_ES_InitializeFileSystems(), CFE_ES_Main(), and CFE_ES_Setup↩ ResetVariables().

13.30.2 Function Documentation

13.30.2.1 CFE_ES_CreateObjects()

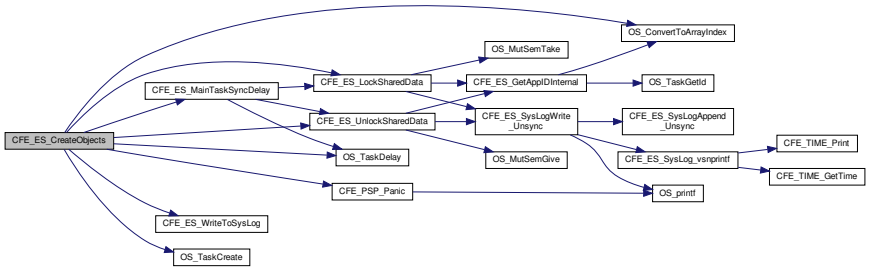
```
void CFE_ES_CreateObjects (
            void  )
```

Definition at line 740 of file cfe_es_start.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_EARLY_INIT, CFE_ES_AppState_RUNNING, CFE_ES_AppState_UNDEFINED, CFE_ES_AppType_CORE, CFE_ES_CORE_TASK, CFE_ES_DRIVER_TASK, CFE_ES_ExceptionAction_PROC_RESTART, CFE_ES_FUNCTION_CALL, CFE_ES_LockSharedData(), CFE_ES_MainTaskSyncDelay(), CFE_ES_NULL_ENTRY, CFE_ES_ObjectTable, CFE_ES_PANIC_DELAY, CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_OBJECT_TABLE_SIZE, CFE_PSP_Panic(), CFE_PSP_PANIC_CORE_APP, CFE_SUCCESS, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_ObjectTable_t::FuncPtrUnion, CFE_ES_FuncPtrUnion_t::FunctionPtr, CFE_ES_FuncPtrUnion_t::MainAppPtr, CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_MainTaskInfo_t::MainTaskName, CFE_ES_AppStartParams_t::Name, NULL, CFE_ES_ObjectTable_t::ObjectName, CFE_ES_ObjectTable_t::ObjectPriority, CFE_ES_ObjectTable_t::ObjectSize, OS_ConvertToArrayIndex(), OS_FP_ENABLED, OS_MAX_API_NAME, OS_SUCCESS, OS_TaskCreate(), OS_TaskDelay(), CFE_ES_AppStartParams_t::Priority, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredCoreApps, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppStartParams_t::StackSize, CFE_ES_AppStartParams_t::StartAddress, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_TaskRecord_t::TaskName, CFE_ES_Global_t::TaskTable, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.30.2.2 CFE_ES_InitializeFileSystems()

```
void CFE_ES_InitializeFileSystems (
            uint32 StartType )
```

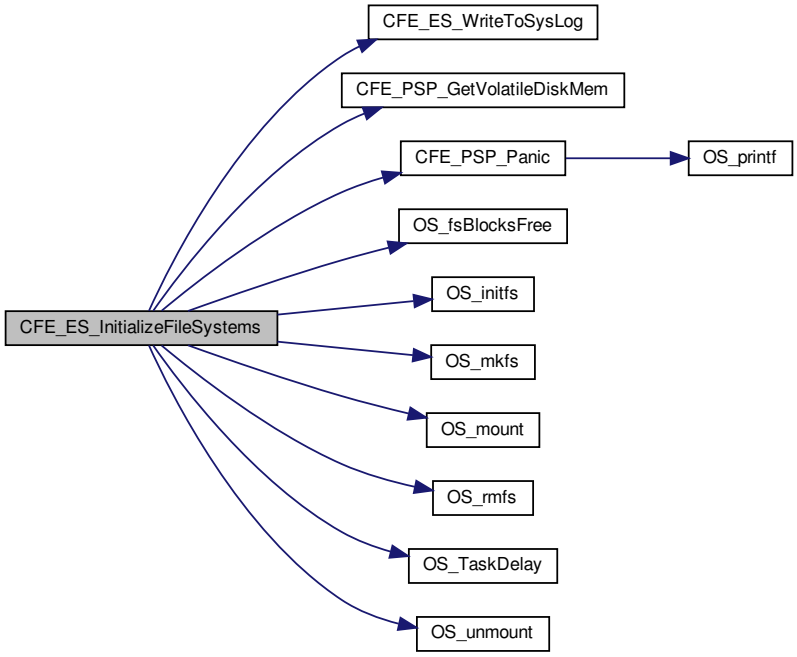
Definition at line 483 of file cfe_es_start.c.

References CFE_ES_PANIC_DELAY, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_RAM_DISK_MOUNT_STRING, CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS, CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED, CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE, CFE_PSP_GetVolatileDiskMem(), CFE_PSP_Panic(),

CFE_PSP_PANIC_VOLATILE_DISK, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_PSP_SUCCESS, OS_FS_SUCCESS, OS_fsBlocksFree(), OS_initfs(), OS_mkfs(), OS_mount(), OS_rmfs(), OS_SUCCESS, OS_TaskDelay(), and OS_unmount().

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.30.2.3 CFE_ES_Main()

```
void CFE_ES_Main (
             uint32 StartType,
             uint32 StartSubtype,
             uint32 ModeId,
             const char * StartFilePath )
```

Description

cFE main entry point. This is the entry point into the cFE software. It is called only by the Board Support Package software.

Assumptions, External Events, and Notes:

None

Parameters

in	*StartType*	Identifies whether this was a CFE_PSP_RST_TYPE_POWERON or CFE_PSP_RST_TYPE_PROCESSOR.
in	*StartSubtype*	Specifies, in more detail, what caused the `StartType` identified above. See CFE_PSP_RST_SUBTYPE_POWER_CYCLE for possible examples.
in	*ModeId*	Identifies the source of the Boot as determined by the BSP.
in	*StartFilePath*	Identifies the startup file to use to initialize the cFE apps.

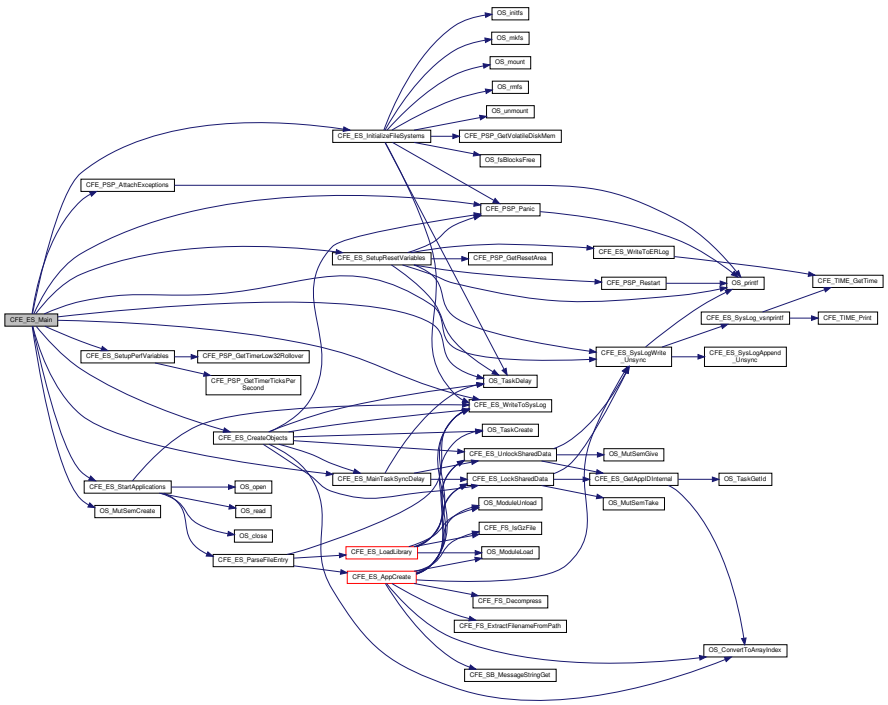
See also

CFE_ES_ResetCFE

Definition at line 86 of file cfe_es_start.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_LATE_INIT, CFE_ES_AppState_RUNNING, CFE_ES_AppState_UNDEFINED, CFE_ES_CreateObjects(), CFE_ES_InitializeFileSystems(), CFE_ES_MainTaskSyncDelay(), CFE_ES_PANIC_DELAY, CFE_ES_SetupPerfVariables(), CFE_ES_SetupResetVariables(), CFE_ES_StartApplications(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_SystemState_APPS_INIT, CFE_ES_SystemState_CORE_READY, CFE_ES_SystemState_CORE_STARTUP, CFE_ES_SystemState_EARLY_INIT, CFE_ES_SystemState_OPERATIONAL, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC, CFE_PSP_AttachExceptions(), CFE_PSP_Panic(), CFE_PSP_PANIC_STARTUP_SEM, CFE_SUCCESS, CFE_ES_Global_t::CounterTable, OS_MAX_TASKS, OS_MutSemCreate(), OS_SUCCESS, OS_TaskDelay(), CFE_ES_GenCounterRecord_t::RecordUsed, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::SharedDataMutex, CFE_ES_Global_t::SystemState, and CFE_ES_Global_t::TaskTable.

Here is the call graph for this function:



13.30.2.4 CFE_ES_MainTaskSyncDelay()

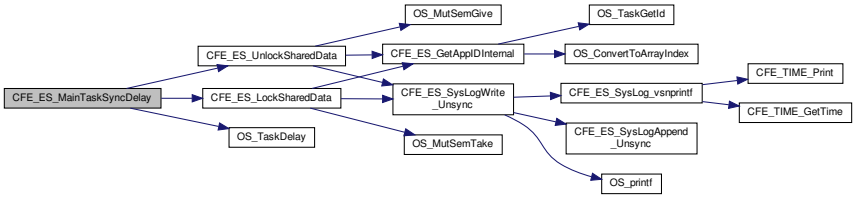
```
int32 CFE_ES_MainTaskSyncDelay (
            uint32 AppStateId,
            uint32 TimeOutMilliseconds )  [static]
```

Definition at line 960 of file cfe_es_start.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_LockSharedData(), CFE_ES_OPERATION_TIMED_OUT, CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC, CFE_SUCCESS, and OS_TaskDelay().

Referenced by CFE_ES_CreateObjects(), and CFE_ES_Main().

Here is the call graph for this function:



13.30.2.5 CFE_ES_SetupResetVariables()

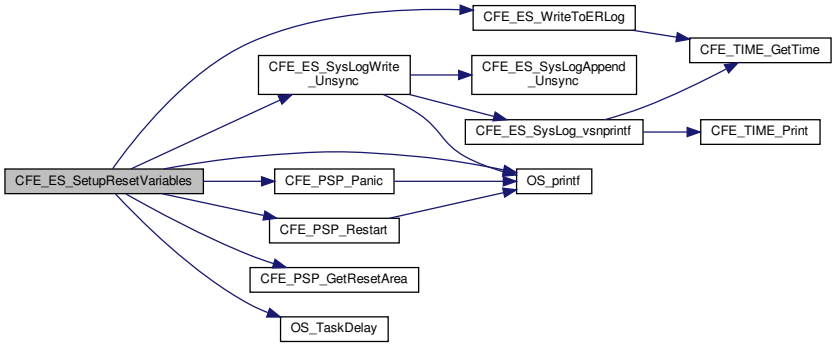
```
void CFE_ES_SetupResetVariables (
            uint32 StartType,
            uint32 StartSubtype,
            uint32 BootSource )
```

Definition at line 249 of file cfe_es_start.c.

References CFE_ES_ResetVariables_t::BootSource, CFE_ES_LogEntryType_CORE, CFE_ES_PANIC_DELAY, CFE_ES_SysLogWrite_Unsync(), CFE_ES_WriteToERLog(), CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS, CFE_PSP_GetResetArea(), CFE_PSP_Panic(), CFE_PSP_PANIC_MEMORY_ALLOC, CFE_PSP_Restart(), CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND, CFE_PSP_RST_SUBTYPE_HW_WATCHDOG, CFE_PSP_RST_SUBTYPE_POWER_CYCLE, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_PSP_SUCCESS, CFE_ES_DebugVariables_t::DebugFlag, CFE_ES_Global_t::DebugVars, CFE_ES_ResetVariables_t::ES_CausedReset, CFE_ES_ResetVariables_t::MaxProcessorResetCount, NULL, OS_printf(), OS_TaskDelay(), CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_ResetVariables_t::ResetSubtype, CFE_ES_ResetVariables_t::ResetType, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.30.3 Variable Documentation

13.30.3.1 CFE_ES_Global

```
CFE_ES_Global_t CFE_ES_Global
```

Definition at line 68 of file cfe_es_start.c.

Referenced by CFE_ES_AppCreate(), CFE_ES_CDS_EarlyInit(), CFE_ES_CDSBlockRead(), CFE_ES_CDSBlock↩ Write(), CFE_ES_CleanUpApp(), CFE_ES_CleanupTaskResources(), CFE_ES_CopyToCDS(), CFE_ES_Create↩ ChildTask(), CFE_ES_DeleteApp(), CFE_ES_DeleteCDS(), CFE_ES_DeleteChildTask(), CFE_ES_DeleteGen↩ Counter(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_FindCD↩ SInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), CFE_ES_GetAppIDByName(), CFE_ES_GetAppIDInternal(), CFE_ES_GetAppInfo(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetAppName(), CFE_ES_GetGenCount(), CF↩ E_ES_GetGenCounterIDByName(), CFE_ES_GetTaskInfo(), CFE_ES_HousekeepingCmd(), CFE_ES_Increment↩ GenCounter(), CFE_ES_IncrementTaskCounter(), CFE_ES_InitCDSRegistry(), CFE_ES_InitializeCDS(), CFE_ES↩ _ListApplications(), CFE_ES_ListTasks(), CFE_ES_LoadLibrary(), CFE_ES_LockCDSRegistry(), CFE_ES_Lock↩ SharedData(), CFE_ES_ProcessControlRequest(), CFE_ES_ProcessCoreException(), CFE_ES_PutCDSBlock(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_RebuildCDS(), CFE_ES_RegisterCDS(), C↩ FE_ES_RegisterCDSEx(), CFE_ES_RegisterGenCounter(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CF↩ E_ES_RestoreFromCDS(), CFE_ES_RunLoop(), CFE_ES_ScanAppTable(), CFE_ES_SetAppState(), CFE_ES↩ _SetGenCount(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UnlockSharedData(), CFE_ES_UpdateCDSRegistry(), CFE_ES_ValidateCDS(), CFE_ES_WaitForSystemState(), and CFE_ES_WriteToERLog().

13.30.3.2 CFE_ES_ResetDataPtr

```
CFE_ES_ResetData_t* CFE_ES_ResetDataPtr
```

Definition at line 73 of file cfe_es_start.c.

Referenced by CFE_ES_ClearERLogCmd(), CFE_ES_GetResetType(), CFE_ES_HousekeepingCmd(), CFE_ES↩ _ProcessCoreException(), CFE_ES_ResetCFE(), CFE_ES_ResetPRCountCmd(), CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetupPerfVariables(), CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogClear_Unsync(), CFE_ES↩ _SysLogReadData(), CFE_ES_SysLogReadStart_Unsync(), CFE_ES_SysLogSetMode(), CFE_ES_TaskInit(), and CFE_ES_WriteToERLog().

13.31 cfe/fsw/cfe-core/src/es/cfe_es_start.h File Reference

```
#include "cfe.h"
```

Data Structures

- union CFE_ES_FuncPtrUnion_t
- struct CFE_ES_ObjectTable_t

Macros

- #define CFE_ES_NULL_ENTRY 0x00
- #define CFE_ES_CORE_TASK 0x01
- #define CFE_ES_DRIVER_TASK 0x02
- #define CFE_ES_BIN_SEM 0x03
- #define CFE_ES_FUNCTION_CALL 0x04
- #define CFE_ES_MUTEX_SEM 0x05

Typedefs

- typedef int32(∗ CFE_ES_EarlyInitFuncPtr_t) (void)

  Req'd prototype of Early Init Functions.
- typedef void(∗ CFE_ES_MainAppFuncPtr_t) (void)

  Req'd prototype of Application Main Functions.

Functions

- void CFE_ES_CreateObjects (void)
- void CFE_ES_SetupResetVariables (uint32 StartType, uint32 StartSubtype, uint32 BootSource)
- void CFE_ES_InitializeFileSystems (uint32 StartType)
- void CFE_ES_SetupPerfVariables (uint32 ResetType)

Variables

- CFE_ES_ObjectTable_t CFE_ES_ObjectTable [CFE_PLATFORM_ES_OBJECT_TABLE_SIZE]

13.31.1 Macro Definition Documentation

13.31.1.1 CFE_ES_BIN_SEM

```
#define CFE_ES_BIN_SEM 0x03
```

Definition at line 56 of file cfe_es_start.h.

13.31.1.2 CFE_ES_CORE_TASK

```
#define CFE_ES_CORE_TASK 0x01
```

Definition at line 54 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

13.31.1.3 CFE_ES_DRIVER_TASK

```
#define CFE_ES_DRIVER_TASK 0x02
```

Definition at line 55 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

13.31.1.4 CFE_ES_FUNCTION_CALL

```
#define CFE_ES_FUNCTION_CALL 0x04
```

Definition at line 57 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

13.31.1.5 CFE_ES_MUTEX_SEM

```
#define CFE_ES_MUTEX_SEM 0x05
```

Definition at line 58 of file cfe_es_start.h.

13.31.1.6 CFE_ES_NULL_ENTRY

```
#define CFE_ES_NULL_ENTRY 0x00
```

Definition at line 53 of file cfe_es_start.h.

Referenced by CFE_ES_CreateObjects().

13.31.2 Typedef Documentation

13.31.2.1 CFE_ES_EarlyInitFuncPtr_t

```
typedef int32(* CFE_ES_EarlyInitFuncPtr_t) (void)
```

Definition at line 64 of file cfe_es_start.h.

13.31.2.2 CFE_ES_MainAppFuncPtr_t

```
typedef void(* CFE_ES_MainAppFuncPtr_t) (void)
```

Definition at line 65 of file cfe_es_start.h.

13.31.3 Function Documentation

13.31.3.1 CFE_ES_CreateObjects()

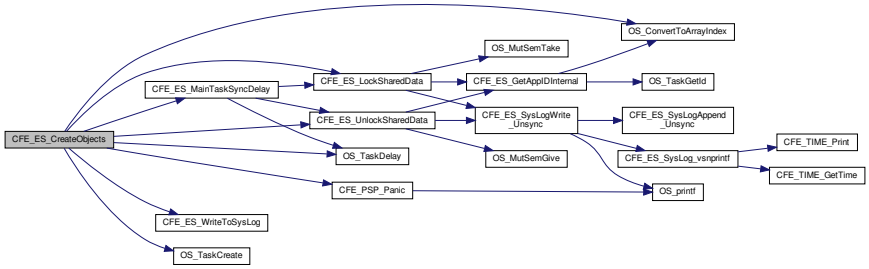
```
void CFE_ES_CreateObjects (
            void  )
```

Definition at line 740 of file cfe_es_start.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_EARLY_INIT, CFE_ES_AppState_RUNNING, CFE_ES_AppState_UNDEFINED, CFE_ES_AppType_CORE, CFE_ES_CORE_TASK, CFE_ES_DRIVER_TASK, CFE_ES_ExceptionAction_PROC_RESTART, CFE_ES_FUNCTION_CALL, CFE_ES_LockSharedData(), CFE_ES_MainTaskSyncDelay(), CFE_ES_NULL_ENTRY, CFE_ES_ObjectTable, CFE_ES_PANIC_DELAY, CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_OBJECT_TABLE_SIZE, CFE_PSP_Panic(), CFE_PSP_PANIC_CORE_APP, CFE_SUCCESS, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_ObjectTable_t::FuncPtrUnion, CFE_ES_FuncPtrUnion_t::FunctionPtr, CFE_ES_FuncPtrUnion_t::MainAppPtr, CFE_ES_MainTaskInfo_t::MainTaskId, CFE_ES_MainTaskInfo_t::MainTaskName, CFE_ES_AppStartParams_t::Name, NULL, CFE_ES_ObjectTable_t::ObjectName, CFE_ES_ObjectTable_t::ObjectPriority, CFE_ES_ObjectTable_t::ObjectSize, OS_ConvertToArrayIndex(), OS_FP_ENABLED, OS_MAX_API_NAME, OS_SUCCESS, OS_TaskCreate(), OS_TaskDelay(), CFE_ES_AppStartParams_t::Priority, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredCoreApps, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppStartParams_t::StackSize, CFE_ES_AppStartParams_t::StartAddress, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_TaskRecord_t::TaskName, CFE_ES_Global_t::TaskTable, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.31.3.2 CFE_ES_InitializeFileSystems()

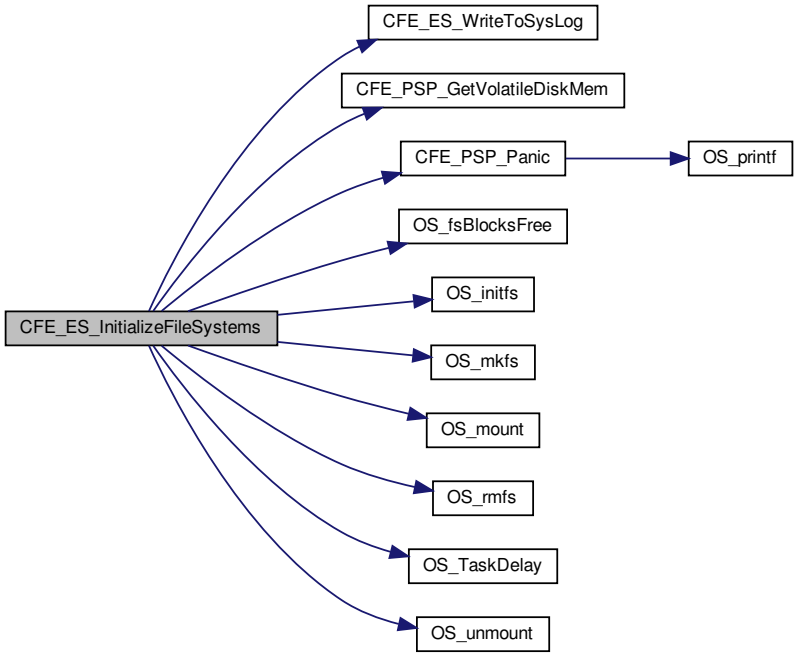
```
void CFE_ES_InitializeFileSystems (
            uint32 StartType )
```

Definition at line 483 of file cfe_es_start.c.

References CFE_ES_PANIC_DELAY, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_RAM_DISK_MOUNT_S↩TRING, CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS, CFE_PLATFORM_ES_RAM_DISK_PERCENT_RE↩SERVED, CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE, CFE_PSP_GetVolatileDiskMem(), CFE_PSP_Panic(), CFE_PSP_PANIC_VOLATILE_DISK, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_PSP_SUCCESS, OS_FS_SUCCESS, OS_fsBlocksFree(), OS_initfs(), OS_mkfs(), OS_mount(), OS_rmfs(), OS_SUCCESS, OS_TaskDelay(), and OS_unmount().

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.31.3.3 CFE_ES_SetupPerfVariables()

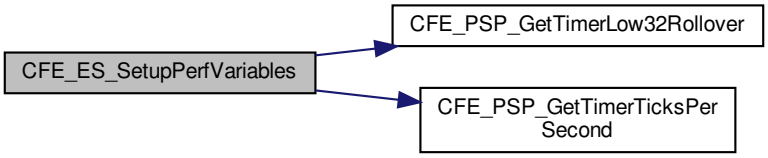
```
void CFE_ES_SetupPerfVariables (
            uint32 ResetType )
```

Definition at line 62 of file cfe_es_perf.c.

References CFE_ES_PERF_32BIT_WORDS_IN_MASK, CFE_ES_PERF_IDLE, CFE_ES_PERF_TRIGGER_START, CFE_ES_ResetDataPtr, CFE_PLATFORM_ES_PERF_FILTMASK_INIT, CFE_PLATFORM_ES_PERF_TRIGMASK_INIT, CFE_PSP_GetTimerLow32Rollover(), CFE_PSP_GetTimerTicksPerSecond(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_ES_PerfLogDump_t::ChildID, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfLogDump_t::DataFileName, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfMetaData_t::Endian, CFE_ES_PerfMetaData_t::FilterMask, CFE_ES_PerfMetaData_t::FilterTriggerMaskSize, CFE_ES_PerfMetaData_t::InvalidMarkerReported, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, CFE_ES_ResetData_t::Perf, CFE_ES_PerfMetaData_t::State, CFE_ES_PerfMetaData_t::TimerLow32Rollover, CFE_ES_PerfMetaData_t::TimerTicksPerSecond, CFE_ES_PerfMetaData_t::TriggerCount, CFE_ES_PerfMetaData_t::TriggerMask, and CFE_ES_PerfMetaData_t::Version.

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.31.3.4 CFE_ES_SetupResetVariables()

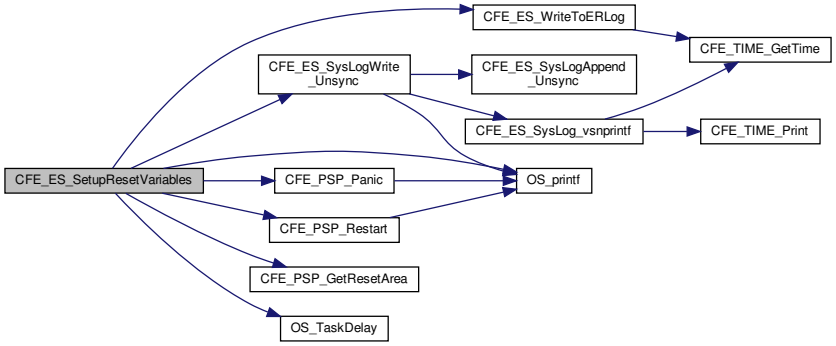
```
void CFE_ES_SetupResetVariables (
            uint32 StartType,
            uint32 StartSubtype,
            uint32 BootSource )
```

Definition at line 249 of file cfe_es_start.c.

References CFE_ES_ResetVariables_t::BootSource, CFE_ES_LogEntryType_CORE, CFE_ES_PANIC_DELAY, CFE_ES_SysLogWrite_Unsync(), CFE_ES_WriteToERLog(), CFE_PLATFORM_ES_MAX_PROCESSOR_RESETS, CFE_PSP_GetResetArea(), CFE_PSP_Panic(), CFE_PSP_PANIC_MEMORY_ALLOC, CFE_PSP_Restart(), CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND, CFE_PSP_RST_SUBTYPE_HW_WATCHDOG, CFE_PSP_RST_SUBTYPE_POWER_CYCLE, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_PSP_SUCCESS, CFE_ES_DebugVariables_t::DebugFlag, CFE_ES_Global_t::DebugVars, CFE_ES_ResetVariables_t::ES_CausedReset, CFE_ES_ResetVariables_t::MaxProcessorResetCount, NULL, OS_printf(), OS_TaskDelay(), CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_ResetVariables_t::ResetSubtype, CFE_ES_ResetVariables_t::ResetType, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_Main().

Here is the call graph for this function:



13.31.4 Variable Documentation

13.31.4.1 CFE_ES_ObjectTable

```
CFE_ES_ObjectTable_t CFE_ES_ObjectTable[CFE_PLATFORM_ES_OBJECT_TABLE_SIZE]
```

Definition at line 49 of file cfe_es_objtab.c.

Referenced by CFE_ES_CreateObjects().

13.32 cfe/fsw/cfe-core/src/es/cfe_es_syslog.c File Reference

```
#include "cfe.h"
#include "cfe_es.h"
#include "cfe_es_global.h"
#include "cfe_es_task.h"
#include "cfe_es_log.h"
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include <ctype.h>
```

Functions

- void CFE_ES_SysLogClear_Unsync (void)

  Clear system log.
- void CFE_ES_SysLogReadStart_Unsync (CFE_ES_SysLogReadBuffer_t *Buffer)

  Begin reading the system log.
- int32 CFE_ES_SysLogAppend_Unsync (const char *LogString)

  Append a complete pre-formatted string to the ES SysLog.
- int32 CFE_ES_SysLogWrite_Unsync (const char *SpecStringPtr,...)

  Write a printf-style formatted string to the system log.
- void CFE_ES_SysLogReadData (CFE_ES_SysLogReadBuffer_t *Buffer)

  Read data from the system log buffer into the local buffer.
- int32 CFE_ES_SysLogSetMode (CFE_ES_LogMode_Enum_t Mode)

  Sets the operating mode of the system log buffer.
- void CFE_ES_SysLog_vsnprintf (char *Buffer, size_t BufferSize, const char *SpecStringPtr, va_list ArgPtr)

  Format a message intended for output to the system log.
- void CFE_ES_SysLog_snprintf (char *Buffer, size_t BufferSize, const char *SpecStringPtr,...)
- int32 CFE_ES_SysLogDump (const char *Filename)

  Write the contents of the syslog to a disk file.

13.32.1 Function Documentation

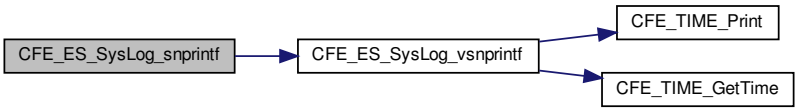
13.32.1.1 CFE_ES_SysLog_snprintf()

```
void CFE_ES_SysLog_snprintf (
            char * Buffer,
            size_t BufferSize,
            const char * SpecStringPtr,
             ...  )
```

Definition at line 454 of file cfe_es_syslog.c.

References CFE_ES_SysLog_vsnprintf().

Here is the call graph for this function:



13.32.1.2 CFE_ES_SysLog_vsnprintf()

```
void CFE_ES_SysLog_vsnprintf (
            char * Buffer,
            size_t BufferSize,
            const char * SpecStringPtr,
            va_list ArgPtr )
```

This function prepares a complete message for passing into CFE_ES_SysLogAppend_Unsync(), based on the given vsnprintf-style specification string and argument list.

The message is prefixed with a time stamp based on the current time, followed by the caller-specified string. An ending newline and terminating null character are both ensured on the output string.

To account for the timestamp, newline, and terminating null character, the supplied buffer must be greater than (CFE↩ _TIME_PRINTED_STRING_SIZE+2) to get a useful output. Any user-specified output string will be truncated to fit into the remaining space.

Parameters

Buffer	User supplied buffer to output formatted sting into
BufferSize	Size of "Buffer" parameter. Should be greater than (CFE_TIME_PRINTED_STRING_SIZE+2)
SpecStringPtr	Printf-style format string
ArgPtr	Variable argument list as obtained by va_start() in the caller

See also

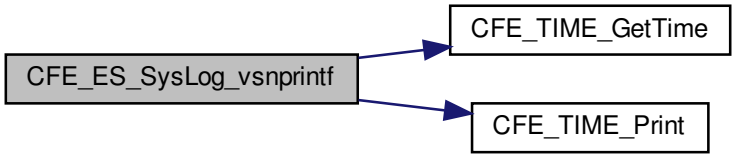
CFE_ES_SysLogAppend_Unsync()

Definition at line 376 of file cfe_es_syslog.c.

References CFE_TIME_GetTime(), CFE_TIME_Print(), and CFE_TIME_PRINTED_STRING_SIZE.

Referenced by CFE_ES_SysLog_snprintf(), CFE_ES_SysLogWrite_Unsync(), and CFE_ES_WriteToSysLog().

Here is the call graph for this function:



13.32.1.3 CFE_ES_SysLogAppend_Unsync()

```
int32 CFE_ES_SysLogAppend_Unsync (
            const char * LogString )
```

The new message will be copied to the current write location in the system log buffer. If there is not sufficient space to completely store the message, then the behavior depends on the "LogMode" setting.

If "LogMode" is set to DISCARD, then the message will be truncated to fit in the available space, or completely discarded if no space exists.

If "LogMode" is set to OVERWRITE, then the oldest message(s) in the system log will be overwritten with this new message.

Parameters

LogString	Message to append

Note

This function requires external thread synchronization

See also

CFE_ES_SysLogSetMode()

Definition at line 143 of file cfe_es_syslog.c.

References CFE_ES_ERR_SYS_LOG_FULL, CFE_ES_ERR_SYS_LOG_TRUNCATED, CFE_ES_LogMode_OVERWRITE, CFE_ES_ResetDataPtr, CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, CFE_SUCCESS, CFE_TIME_PRINTED_STRING_SIZE, CFE_ES_ResetData_t::SystemLog, CFE_ES_ResetData_t::SystemLogEndIdx, CFE_ES_ResetData_t::SystemLogEntryNum, CFE_ES_ResetData_t::SystemLogMode, and CFE_ES_ResetData_t::SystemLogWriteIdx.

Referenced by CFE_ES_SysLogWrite_Unsync(), and CFE_ES_WriteToSysLog().

13.32.1.4 CFE_ES_SysLogClear_Unsync()

```
void CFE_ES_SysLogClear_Unsync (
            void  )
```

This discards the entire system log buffer and resets internal index values

Note

This function requires external thread synchronization

Definition at line 88 of file cfe_es_syslog.c.

References CFE_ES_ResetDataPtr, CFE_ES_ResetData_t::SystemLogEndIdx, CFE_ES_ResetData_t::SystemLogEntryNum, and CFE_ES_ResetData_t::SystemLogWriteIdx.

Referenced by CFE_ES_ClearSyslogCmd().

13.32.1.5 CFE_ES_SysLogDump()

```
int32 CFE_ES_SysLogDump (
            const char * Filename )
```

Writes the current contents of the syslog buffer to a file specified by the Filename parameter. The log messages will be written to the file in the same order in which they were written into the syslog buffer.

A snapshot of the log indices is taken at the beginning of the writing process. Additional log entries added after this (e.g. from applications calling CFE_ES_WriteToSyslog() after starting a syslog dump) will not be included in the dump file.

Note that preference is given to the realtime application threads over any pending log read activities, such as a dumping to a file. The design of this function can tolerate a limited level of logging activity while the dump is in progress without any negative side effects. However, a significant "flood" of log messages may corrupt the output file, by overwriting older data before it has actually been written.

Parameters

Filename	Output file to write

Returns

CFE_SUCCESS if successful, or an appropriate error code from cfe_error.h

See also

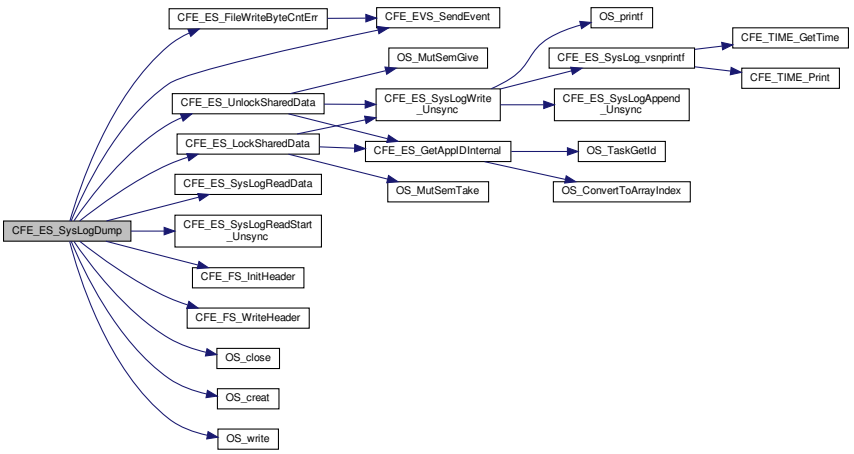
CFE_ES_SYSLOG_READ_BUFFER_SIZE

Definition at line 469 of file cfe_es_syslog.c.

References CFE_ES_FILE_IO_ERR, CFE_ES_FileWriteByteCntErr(), CFE_ES_LockSharedData(), CFE_ES_SYS↩
_LOG_DESC, CFE_ES_SYSLOG2_EID, CFE_ES_SYSLOG2_ERR_EID, CFE_ES_SysLogReadData(), CFE_ES_↩
SysLogReadStart_Unsync(), CFE_ES_TaskData, CFE_ES_UnlockSharedData(), CFE_EVS_EventType_DEBUG, C↩
FE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_SYSLOG, C↩
FE_FS_WriteHeader(), CFE_SUCCESS, CFE_ES_TaskData_t::HkPacket, OS_close(), OS_creat(), OS_write(), OS_↩
WRITE_ONLY, CFE_ES_HousekeepingTlm_t::Payload, and CFE_ES_HousekeepingTlm_Payload_t::SysLogEntries.

Referenced by CFE_ES_WriteSyslogCmd().

Here is the call graph for this function:



13.32.1.6 CFE_ES_SysLogReadData()

```
void CFE_ES_SysLogReadData (
            CFE_ES_SysLogReadBuffer_t * Buffer )
```

Prior to calling this function, the buffer structure should be initialized using CFE_ES_SysLogReadStart_Unsync()

This copies the data from the syslog memory space into the local buffer, starting from the end of the previously read data. To read the complete system log, this function should be called repeatedly until the "BlockSize" member in the returned buffer is returned as zero, indicating there is no more data in the syslog.

There is no specific external synchronization requirement on this function, since copies of the relevant log indices are kept in the buffer structure itself. However, if system log data is overwritten between calls to this function, it may result in undefined data being returned to the caller.

Therefore, in cases where it is critically important to read log message data, the lock should be held for the entire procedure (initialization through complete read). However this may have significant realtime implications, so it is not the required mode of operation.

Parameters

Buffer	A local buffer which will be filled with data from the log buffer

Definition at line 302 of file cfe_es_syslog.c.

References CFE_ES_SysLogReadBuffer_t::BlockSize, CFE_ES_ResetDataPtr, CFE_ES_SysLogReadBuffer_t::Data, CFE_ES_SysLogReadBuffer_t::EndIdx, CFE_ES_SysLogReadBuffer_t::LastOffset, CFE_ES_SysLogReadBuffer_t::↩ SizeLeft, and CFE_ES_ResetData_t::SystemLog.

Referenced by CFE_ES_SysLogDump().

13.32.1.7 CFE_ES_SysLogReadStart_Unsync()

```
void CFE_ES_SysLogReadStart_Unsync (
            CFE_ES_SysLogReadBuffer_t * Buffer )
```

This a helper function is intended to assist with the "Write" command to dump the contents of the syslog to a disk file. This locates the oldest complete log message currently contained in the buffer.

The oldest log message may be overwritten when any application calls CFE_ES_WriteToSysLog() if set to OVERWRITE mode.

This function only locates the first message, it does not actually copy any data to the supplied buffer. The CFE_ES_↩ SysLogReadData() should be called to read log data.

Parameters

Buffer	A local buffer which will be initialized to the start of the log buffer

Note

This function requires external thread synchronization

See also

CFE_ES_SysLogReadData()

Definition at line 107 of file cfe_es_syslog.c.

References CFE_ES_SysLogReadBuffer_t::BlockSize, CFE_ES_ResetDataPtr, CFE_ES_SysLogReadBuffer_t::EndIdx, CFE_ES_SysLogReadBuffer_t::LastOffset, CFE_ES_SysLogReadBuffer_t::SizeLeft, CFE_ES_ResetData_t::SystemLog, CFE_ES_ResetData_t::SystemLogEndIdx, and CFE_ES_ResetData_t::SystemLogWriteIdx.

Referenced by CFE_ES_SysLogDump().

13.32.1.8 CFE_ES_SysLogSetMode()

```
int32 CFE_ES_SysLogSetMode (
            CFE_ES_LogMode_Enum_t Mode )
```

The operating mode of the system log controls its behavior once filled to the point where additional messages can no longer be stored.

If "Mode" is set to DISCARD, then the message will be truncated to fit in the available space, or completely discarded if no space exists.

If "Mode" is set to OVERWRITE, then the oldest message(s) in the system log will be overwritten with this new message.

Note

Switching from OVERWRITE to DISCARD mode may take effect immediately, as the setting only takes effect when the buffer "wrap-point" is reached at the end.

Parameters

Mode	The desired operating mode

Returns

CFE_SUCCESS if set successfully

Definition at line 352 of file cfe_es_syslog.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_LogMode_DISCARD, CFE_ES_LogMode_OVERWRITE, CFE_ES_ResetDataPtr, CFE_SUCCESS, and CFE_ES_ResetData_t::SystemLogMode.

Referenced by CFE_ES_OverWriteSyslogCmd().

13.32.1.9 CFE_ES_SysLogWrite_Unsync()

```
int32 CFE_ES_SysLogWrite_Unsync (
            const char * SpecStringPtr,
             ...  )
```

This is a drop-in replacement for the existing CFE_ES_WriteToSysLog() API that does *not* perform any synchronization or locking. It is intended for logging from within the ES subsystem where the appropriate lock is already held for other reasons.

Note

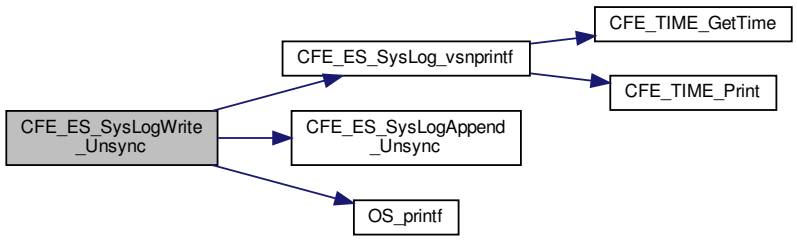
This function requires external thread synchronization

Definition at line 267 of file cfe_es_syslog.c.

References CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_SysLog_vsnprintf(), CFE_ES_SysLogAppend_Unsync(), and OS_printf().

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CleanupObjectCallback(), CFE_ES_↩ CreateCDSPool(), CFE_ES_CreateChildTask(), CFE_ES_DeleteApp(), CFE_ES_DeleteChildTask(), CFE_ES_Exit↩ App(), CFE_ES_ExitChildTask(), CFE_ES_GetTaskInfo(), CFE_ES_LockSharedData(), CFE_ES_Main(), CFE_ES↩ _RebuildCDSPool(), CFE_ES_ReloadApp(), CFE_ES_RestartApp(), CFE_ES_RunLoop(), CFE_ES_SetupReset↩ Variables(), and CFE_ES_UnlockSharedData().

Here is the call graph for this function:



13.33 cfe/fsw/cfe-core/src/es/cfe_es_task.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_platform_cfg.h"
#include "cfe_version.h"
#include "cfe_es_global.h"
#include "cfe_es_apps.h"
#include "cfe_es_events.h"
#include "cfe_es_verify.h"
#include "cfe_es_task.h"
#include "cfe_es_shell.h"
```

```
#include "cfe_es_log.h"
#include "cfe_es_cds.h"
#include "cfe_fs.h"
#include "cfe_psp.h"
#include "cfe_msgids.h"
#include <string.h>
```

Macros

- #define CFE_ES_PERF_TRIGGERMASK_INT_SIZE (sizeof(CFE_ES_ResetDataPtr->Perf.MetaData.TriggerMask) / sizeof(uint32))
- #define CFE_ES_PERF_TRIGGERMASK_EXT_SIZE (sizeof(CFE_ES_TaskData.HkPacket.Payload.PerfTriggerMask) / sizeof(uint32))
- #define CFE_ES_PERF_FILTERMASK_INT_SIZE (sizeof(CFE_ES_ResetDataPtr->Perf.MetaData.FilterMask) / sizeof(uint32))
- #define CFE_ES_PERF_FILTERMASK_EXT_SIZE (sizeof(CFE_ES_TaskData.HkPacket.Payload.PerfFilterMask) / sizeof(uint32))
- #define OS_MAX_PRIORITY 255

Functions

- void CFE_ES_TaskMain (void)

  Entry Point for cFE Core Application.
- int32 CFE_ES_TaskInit (void)
- void CFE_ES_TaskPipe (CFE_SB_MsgPtr_t Msg)
- int32 CFE_ES_HousekeepingCmd (const CCSDS_CommandPacket_t *data)
- int32 CFE_ES_NoopCmd (const CFE_ES_Noop_t *Cmd)
- int32 CFE_ES_ResetCountersCmd (const CFE_ES_ResetCounters_t *data)
- int32 CFE_ES_RestartCmd (const CFE_ES_Restart_t *data)
- int32 CFE_ES_ShellCmd (const CFE_ES_Shell_t *data)
- int32 CFE_ES_StartAppCmd (const CFE_ES_StartApp_t *data)
- int32 CFE_ES_StopAppCmd (const CFE_ES_StopApp_t *data)
- int32 CFE_ES_RestartAppCmd (const CFE_ES_RestartApp_t *data)
- int32 CFE_ES_ReloadAppCmd (const CFE_ES_ReloadApp_t *data)
- int32 CFE_ES_QueryOneCmd (const CFE_ES_QueryOne_t *data)
- int32 CFE_ES_QueryAllCmd (const CFE_ES_QueryAll_t *data)
- int32 CFE_ES_QueryAllTasksCmd (const CFE_ES_QueryAllTasks_t *data)
- int32 CFE_ES_ClearSyslogCmd (const CFE_ES_ClearSyslog_t *data)
- int32 CFE_ES_OverWriteSyslogCmd (const CFE_ES_OverWriteSyslog_t *data)
- int32 CFE_ES_WriteSyslogCmd (const CFE_ES_WriteSyslog_t *data)
- int32 CFE_ES_ClearERLogCmd (const CFE_ES_ClearERLog_t *data)
- int32 CFE_ES_WriteERLogCmd (const CFE_ES_WriteERLog_t *data)
- int32 CFE_ES_ERLogDump (const char *Filename)
- bool CFE_ES_VerifyCmdLength (CFE_SB_MsgPtr_t Msg, uint16 ExpectedLength)
- int32 CFE_ES_ResetPRCountCmd (const CFE_ES_ResetPRCount_t *data)
- int32 CFE_ES_SetMaxPRCountCmd (const CFE_ES_SetMaxPRCount_t *data)
- int32 CFE_ES_DeleteCDSCmd (const CFE_ES_DeleteCDS_t *data)
- int32 CFE_ES_SendMemPoolStatsCmd (const CFE_ES_SendMemPoolStats_t *data)
- int32 CFE_ES_DumpCDSRegistryCmd (const CFE_ES_DumpCDSRegistry_t *data)
- void CFE_ES_FileWriteByteCntErr (const char *Filename, uint32 Requested, uint32 Actual)

Variables

- CFE_ES_TaskData_t CFE_ES_TaskData

13.33.1 Macro Definition Documentation

13.33.1.1 CFE_ES_PERF_FILTERMASK_EXT_SIZE

```
#define CFE_ES_PERF_FILTERMASK_EXT_SIZE (sizeof(CFE_ES_TaskData.HkPacket.Payload.PerfFilterMask) /
sizeof(uint32))
```

Definition at line 62 of file cfe_es_task.c.

Referenced by CFE_ES_HousekeepingCmd().

13.33.1.2 CFE_ES_PERF_FILTERMASK_INT_SIZE

```
#define CFE_ES_PERF_FILTERMASK_INT_SIZE (sizeof(CFE_ES_ResetDataPtr->Perf.MetaData.FilterMask) /
sizeof(uint32))
```

Definition at line 61 of file cfe_es_task.c.

Referenced by CFE_ES_HousekeepingCmd().

13.33.1.3 CFE_ES_PERF_TRIGGERMASK_EXT_SIZE

```
#define CFE_ES_PERF_TRIGGERMASK_EXT_SIZE (sizeof(CFE_ES_TaskData.HkPacket.Payload.PerfTriggerMask)
/ sizeof(uint32))
```

Definition at line 60 of file cfe_es_task.c.

Referenced by CFE_ES_HousekeepingCmd().

13.33.1.4 CFE_ES_PERF_TRIGGERMASK_INT_SIZE

```
#define CFE_ES_PERF_TRIGGERMASK_INT_SIZE (sizeof(CFE_ES_ResetDataPtr->Perf.MetaData.TriggerMask)
/ sizeof(uint32))
```

Definition at line 59 of file cfe_es_task.c.

Referenced by CFE_ES_HousekeepingCmd().

13.33.1.5 OS_MAX_PRIORITY

```
#define OS_MAX_PRIORITY 255
```

Definition at line 67 of file cfe_es_task.c.

Referenced by CFE_ES_StartAppCmd().

13.33.2 Function Documentation

13.33.2.1 CFE_ES_ClearERLogCmd()

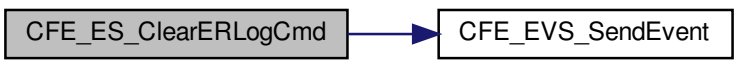
```
int32 CFE_ES_ClearERLogCmd (
            const CFE_ES_ClearERLog_t * data )
```

Definition at line 1536 of file cfe_es_task.c.

References CFE_ES_ERLOG1_INF_EID, CFE_ES_ResetDataPtr, CFE_EVS_EventType_INFORMATION, CFE_EV↩S_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_ResetData_t::ERLog, CFE_E↩S_ResetData_t::ERLogEntries, and CFE_ES_ResetData_t::ERLogIndex.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.2 CFE_ES_ClearSyslogCmd()

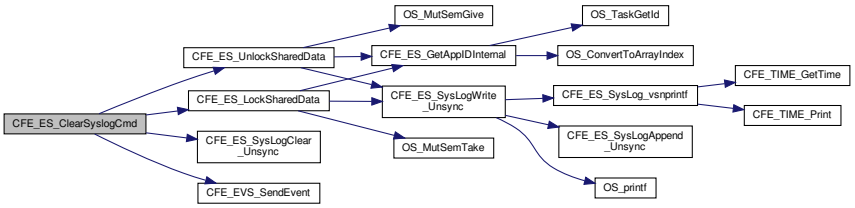
```
int32 CFE_ES_ClearSyslogCmd (
            const CFE_ES_ClearSyslog_t * data )
```

Definition at line 1446 of file cfe_es_task.c.

References CFE_ES_LockSharedData(), CFE_ES_SYSLOG1_INF_EID, CFE_ES_SysLogClear_Unsync(), CFE_ES_UnlockSharedData(), CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, and CFE_ES_TaskData_t::CommandCounter.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.3 CFE_ES_DeleteCDSCmd()

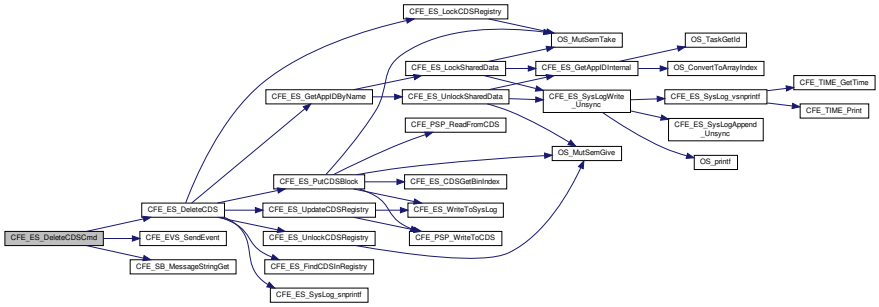
```
int32 CFE_ES_DeleteCDSCmd (
            const CFE_ES_DeleteCDS_t * data )
```

Definition at line 1751 of file cfe_es_task.c.

References CFE_ES_DeleteCDSCmd_Payload_t::CdsName, CFE_ES_CDS_DELETE_ERR_EID, CFE_ES_CDS_DELETE_TBL_ERR_EID, CFE_ES_CDS_DELETED_INFO_EID, CFE_ES_CDS_MAX_FULL_NAME_LEN, CFE_ES_CDS_NAME_ERR_EID, CFE_ES_CDS_NOT_FOUND_ERR, CFE_ES_CDS_OWNER_ACTIVE_EID, CFE_ES_CDS_OWNER_ACTIVE_ERR, CFE_ES_CDS_WRONG_TYPE_ERR, CFE_ES_DeleteCDS(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, and CFE_ES_DeleteCDS_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.4 CFE_ES_DumpCDSRegistryCmd()

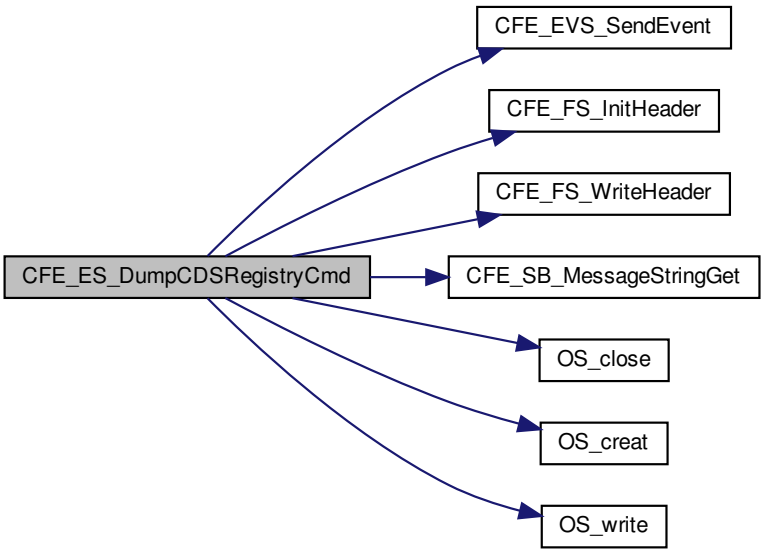
```
int32 CFE_ES_DumpCDSRegistryCmd (
            const CFE_ES_DumpCDSRegistry_t * data )
```

Definition at line 1859 of file cfe_es_task.c.

References CFE_ES_CDSRegDumpRec_t::ByteAlignSpare1, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_DUMP_ERR_EID, CFE_ES_CDS_REG_DUMP_INF_EID, CFE_ES_CREATING_CDS_DUMP_ERR_EID, CFE_ES_Global, CFE_ES_WRITE_CFE_HDR_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_CDS_REG, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_DumpCDSRegistryCmd_Payload_t::DumpFilename, CFE_ES_CDSRegDumpRec_t::Handle, CFE_ES_CDS_RegRec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, CFE_ES_CDSRegDumpRec_t::Name, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_MAX_PATH_LEN, OS_write(), OS_WRITE_ONLY, CFE_ES_DumpCDSRegistry_t::Payload, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec_t::Size, CFE_ES_CDSRegDumpRec_t::Size, CFE_ES_CDS_RegRec_t::Table, CFE_ES_CDSRegDumpRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.5 CFE_ES_ERLogDump()

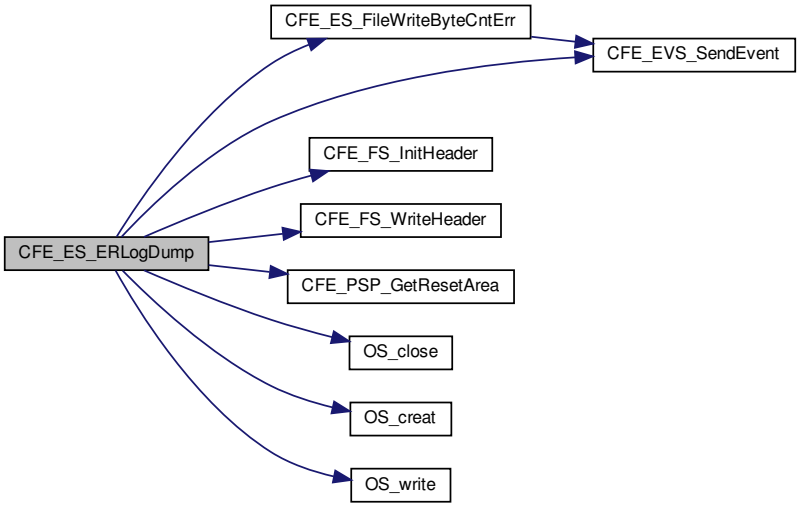
```
int32 CFE_ES_ERLogDump (
            const char * Filename )
```

Definition at line 1600 of file cfe_es_task.c.

References CFE_ES_ER_LOG_DESC, CFE_ES_ERLOG2_EID, CFE_ES_ERLOG2_ERR_EID, CFE_ES_FILE_IO_ERR, CFE_ES_FileWriteByteCntErr(), CFE_ES_RST_ACCESS_EID, CFE_ES_RST_ACCESS_ERR, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_ERLOG, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_ER_LOG_ENTRIES, CFE_PSP_GetResetArea(), CFE_PSP_SUCCESS, CFE_SUCCESS, OS_close(), OS_creat(), OS_write(), and OS_WRITE_ONLY.

Referenced by CFE_ES_WriteERLogCmd().

Here is the call graph for this function:



13.33.2.6 CFE_ES_FileWriteByteCntErr()

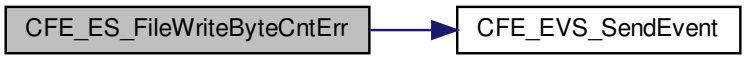
```
void CFE_ES_FileWriteByteCntErr (
            const char * Filename,
            uint32 Requested,
            uint32 Actual )
```

Definition at line 1978 of file cfe_es_task.c.

References CFE_ES_FILEWRITE_ERR_EID, CFE_EVS_EventType_ERROR, and CFE_EVS_SendEvent().

Referenced by CFE_ES_ERLogDump(), CFE_ES_PerfLogDump(), and CFE_ES_SysLogDump().

Here is the call graph for this function:



13.33.2.7 CFE_ES_HousekeepingCmd()

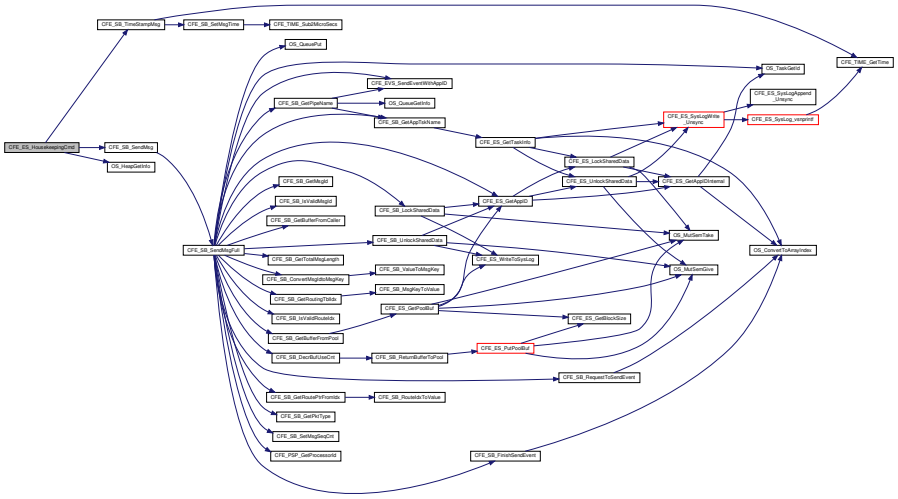
```
int32 CFE_ES_HousekeepingCmd (
            const CCSDS_CommandPacket_t * data )
```

Definition at line 638 of file cfe_es_task.c.

References CFE_ES_ResetVariables_t::BootSource, CFE_ES_HousekeepingTlm_Payload_t::BootSource, CFE_ES_Global, CFE_ES_PERF_FILTERMASK_EXT_SIZE, CFE_ES_PERF_FILTERMASK_INT_SIZE, CFE_ES_PERF_TRIGGERMASK_EXT_SIZE, CFE_ES_PERF_TRIGGERMASK_INT_SIZE, CFE_ES_PerfLogDumpStatus, CFE_ES_ResetDataPtr, CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_HousekeepingTlm_Payload_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_HousekeepingTlm_Payload_t::CommandErrorCounter, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_ResetData_t::ERLogEntries, CFE_ES_HousekeepingTlm_Payload_t::ERLogEntries, CFE_ES_ResetData_t::ERLogIndex, CFE_ES_HousekeepingTlm_Payload_t::ERLogIndex, CFE_ES_PerfMetaData_t::FilterMask, OS_heap_prop_t::free_blocks, OS_heap_prop_t::free_bytes, CFE_ES_HousekeepingTlm_Payload_t::HeapBlocksFree, CFE_ES_HousekeepingTlm_Payload_t::HeapBytesFree, CFE_ES_HousekeepingTlm_Payload_t::HeapMaxBlockSize, CFE_ES_TaskData_t::HkPacket, OS_heap_prop_t::largest_free_block, CFE_ES_ResetVariables_t::MaxProcessorResetCount, CFE_ES_HousekeepingTlm_Payload_t::MaxProcessorResets, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, OS_HeapGetInfo(), OS_SUCCESS, CFE_ES_HousekeepingTlm_t::Payload, CFE_ES_ResetData_t::Perf, CFE_ES_HousekeepingTlm_Payload_t::PerfDataCount, CFE_ES_HousekeepingTlm_Payload_t::PerfDataEnd, CFE_ES_HousekeepingTlm_Payload_t::PerfDataStart, CFE_ES_HousekeepingTlm_Payload_t::PerfDataToWrite, CFE_ES_HousekeepingTlm_Payload_t::PerfFilterMask, CFE_ES_HousekeepingTlm_Payload_t::PerfMode, CFE_ES_HousekeepingTlm_Payload_t::PerfState, CFE_ES_HousekeepingTlm_Payload_t::PerfTriggerCount, CFE_ES_HousekeepingTlm_Payload_t::PerfTriggerMask, CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_HousekeepingTlm_Payload_t::ProcessorResets, CFE_ES_Global_t::RegisteredCoreApps, CFE_ES_HousekeepingTlm_Payload_t::RegisteredCoreApps, CFE_ES_Global_t::RegisteredExternalApps, CFE_ES_HousekeepingTlm_Payload_t::RegisteredExternalApps, CFE_ES_Global_t::RegisteredLibs, CFE_ES_HousekeepingTlm_Payload_t::RegisteredLibs, CFE_ES_Global_t::RegisteredTasks, CFE_ES_HousekeepingTlm_Payload_t::RegisteredTasks, CFE_ES_ResetVariables_t::ResetSubtype, CFE_ES_HousekeepingTlm_Payload_t::ResetSubtype, CFE_ES_ResetVariables_t::ResetType, CFE_ES_HousekeepingTlm_Payload_t::ResetType, CFE_ES_ResetData_t::ResetVars, CFE_ES_PerfMetaData_t::State, CFE_ES_HousekeepingTlm_Payload_t::SysLogBytesUsed, CFE_ES_HousekeepingTlm_Payload_t::SysLogEntries, CFE_ES_HousekeepingTlm_Payload_t::SysLogMode, CFE_ES_HousekeepingTlm_Payload_t::SysLogSize, CFE_ES_ResetData_t::SystemLogEndIdx, CFE_ES_ResetData_t::SystemLogEntryNum, CFE_ES_ResetData_t::SystemLogMode, CFE_ES_PerfMetaData_t::TriggerCount, and CFE_ES_PerfMetaData_t::TriggerMask.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.8 CFE_ES_NoopCmd()

```
int32 CFE_ES_NoopCmd (
            const CFE_ES_Noop_t * Cmd )
```

Definition at line 748 of file cfe_es_task.c.

References CFE_ES_BUILD_INF_EID, CFE_ES_NOOP_INF_EID, CFE_EVS_EventType_INFORMATION, CFE_E↩VS_SendEvent(), CFE_MAJOR_VERSION, CFE_MINOR_VERSION, CFE_MISSION_REV, CFE_PSP_MAJOR_VE↩RSION, CFE_PSP_MINOR_VERSION, CFE_PSP_MISSION_REV, CFE_PSP_REVISION, CFE_REVISION, CFE_↩SUCCESS, CFE_ES_TaskData_t::CommandCounter, OS_MAJOR_VERSION, OS_MINOR_VERSION, OS_MISSIO↩N_REV, and OS_REVISION.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:

13.33.2.9 CFE_ES_OverWriteSyslogCmd()

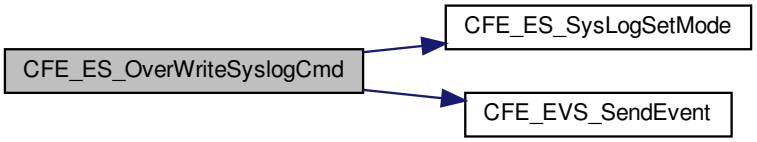
```
int32 CFE_ES_OverWriteSyslogCmd (
            const CFE_ES_OverWriteSyslog_t * data )
```

Definition at line 1472 of file cfe_es_task.c.

References CFE_ES_ERR_SYSLOGMODE_EID, CFE_ES_SYSLOGMODE_EID, CFE_ES_SysLogSetMode(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_OverWriteSysLogCmd_Payload_t::Mode, and CFE_ES_OverWriteSyslog_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.10 CFE_ES_QueryAllCmd()

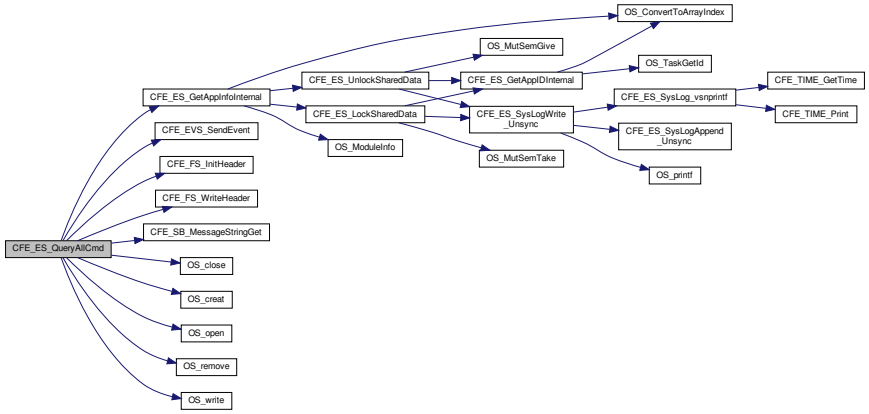
```
int32 CFE_ES_QueryAllCmd (
            const CFE_ES_QueryAll_t * data )
```

Definition at line 1194 of file cfe_es_task.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ALL_APPS_EID, CFE_ES_APP_LOG_DESC, CFE_ES_AppState_UNDEFINED, CFE_ES_GetAppInfoInternal(), CFE_ES_Global, CFE_ES_OSCREATE_ERR_EID, CFE_ES_TASKWR_ERR_EID, CFE_ES_WRHDR_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_QUERYALL, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_close(), OS_creat(), OS_MAX_PATH_LEN, OS_open(), OS_READ_ONLY, OS_remove(), OS_write(), OS_WRITE_ONLY, and CFE_ES_FileNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.11 CFE_ES_QueryAllTasksCmd()

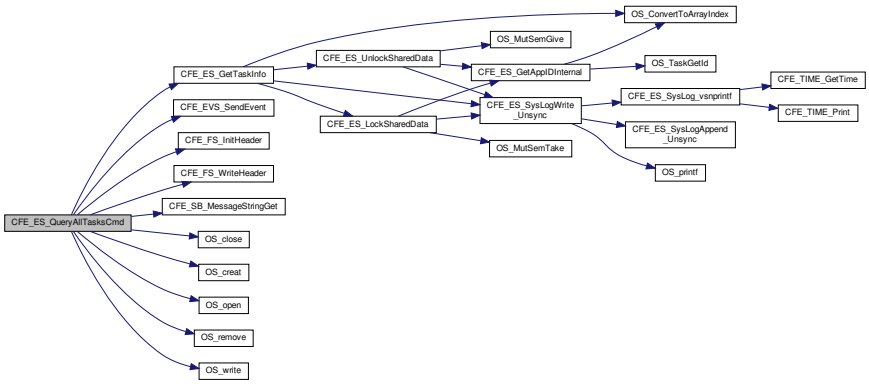
```
int32 CFE_ES_QueryAllTasksCmd (
            const CFE_ES_QueryAllTasks_t * data )
```

Definition at line 1319 of file cfe_es_task.c.

References CFE_ES_GetTaskInfo(), CFE_ES_Global, CFE_ES_TASK_LOG_DESC, CFE_ES_TASKINFO_EID, CFE_ES_TASKINFO_OSCREATE_ERR_EID, CFE_ES_TASKINFO_WR_ERR_EID, CFE_ES_TASKINFO_WRHDR_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_QUERYALLTASKS, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_close(), OS_creat(), OS_MAX_PATH_LEN, OS_MAX_TASKS, OS_open(), OS_READ_ONLY, OS_remove(), OS_write(), OS_WRITE_ONLY, CFE_ES_FileNameCmd_t::Payload, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_TaskRecord_t::TaskId, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.12 CFE_ES_QueryOneCmd()

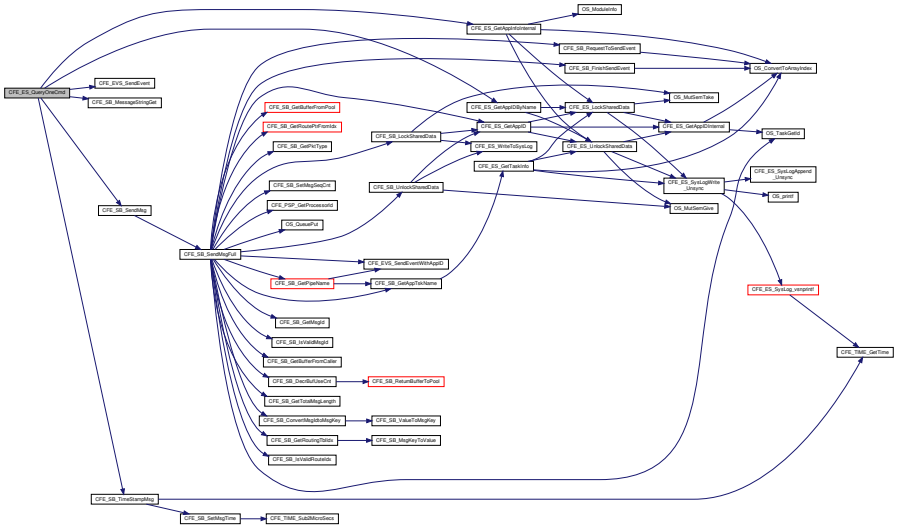
```
int32 CFE_ES_QueryOneCmd (
            const CFE_ES_QueryOne_t * data )
```

Definition at line 1140 of file cfe_es_task.c.

References CFE_ES_OneAppTlm_Payload_t::AppInfo, CFE_ES_AppNameCmd_Payload_t::Application, CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_ONE_APP_EID, CFE_ES_ONE_APPID_ERR_EID, CFE_ES_ONE_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, CFE_ES_TaskData_t::OneAppPacket, OS_MAX_API_NAME, CFE_ES_AppNameCmd_t::Payload, and CFE_ES_OneAppTlm_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.13 CFE_ES_ReloadAppCmd()

```
int32 CFE_ES_ReloadAppCmd (
            const CFE_ES_ReloadApp_t * data )
```

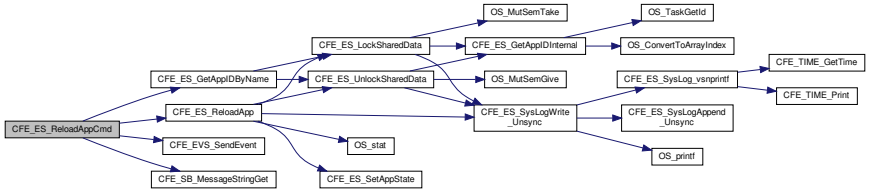
Definition at line 1087 of file cfe_es_task.c.

References CFE_ES_AppReloadCmd_Payload_t::AppFileName, CFE_ES_AppReloadCmd_Payload_t::Application, CFE_ES_GetAppIDByName(), CFE_ES_RELOAD_APP_DBG_EID, CFE_ES_RELOAD_APP_ERR1_EID, CFE_ES_RELOAD_APP_ERR2_EID, CFE_ES_ReloadApp(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR,

CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, OS_MAX_API_NAME, OS_MAX_PATH_LEN, and CFE_ES_↩ ReloadApp_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.14 CFE_ES_ResetCountersCmd()

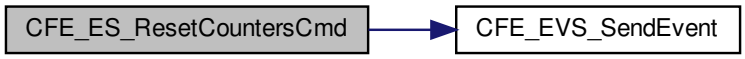
```
int32 CFE_ES_ResetCountersCmd (
            const CFE_ES_ResetCounters_t * data )
```

Definition at line 780 of file cfe_es_task.c.

References CFE_ES_RESET_INF_EID, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SU↩ CCESS, CFE_ES_TaskData_t::CommandCounter, and CFE_ES_TaskData_t::CommandErrorCounter.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.15 CFE_ES_ResetPRCountCmd()

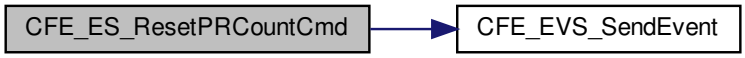
```
int32 CFE_ES_ResetPRCountCmd (
            const CFE_ES_ResetPRCount_t * data )
```

Definition at line 1701 of file cfe_es_task.c.

References CFE_ES_RESET_PR_COUNT_EID, CFE_ES_ResetDataPtr, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_ResetVariables_t::↩ ProcessorResetCount, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.16 CFE_ES_RestartAppCmd()

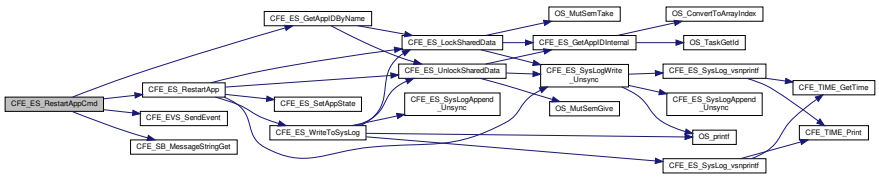
```
int32 CFE_ES_RestartAppCmd (
            const CFE_ES_RestartApp_t * data )
```

Definition at line 1038 of file cfe_es_task.c.

References CFE_ES_AppNameCmd_Payload_t::Application, CFE_ES_GetAppIDByName(), CFE_ES_RESTART_A↩ PP_DBG_EID, CFE_ES_RESTART_APP_ERR1_EID, CFE_ES_RESTART_APP_ERR2_EID, CFE_ES_RestartApp(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageString↩ Get(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, N↩ ULL, OS_MAX_API_NAME, and CFE_ES_AppNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.17 CFE_ES_RestartCmd()

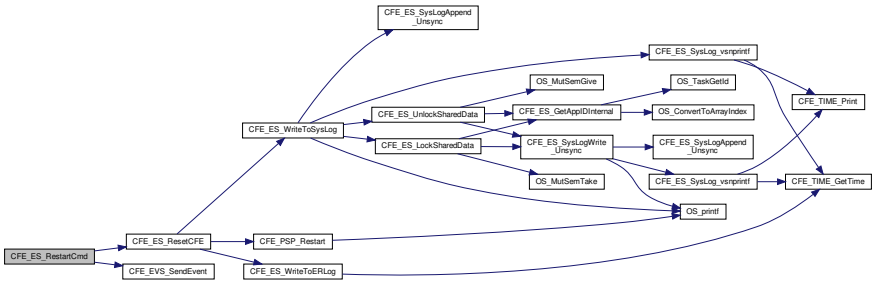
```
int32 CFE_ES_RestartCmd (
            const CFE_ES_Restart_t * data )
```

Definition at line 801 of file cfe_es_task.c.

References CFE_ES_BOOT_ERR_EID, CFE_ES_ResetCFE(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_SUCCESS, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_Restart_t::Payload, and CFE_ES_RestartCmd_Payload_t::RestartType.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.18 CFE_ES_SendMemPoolStatsCmd()

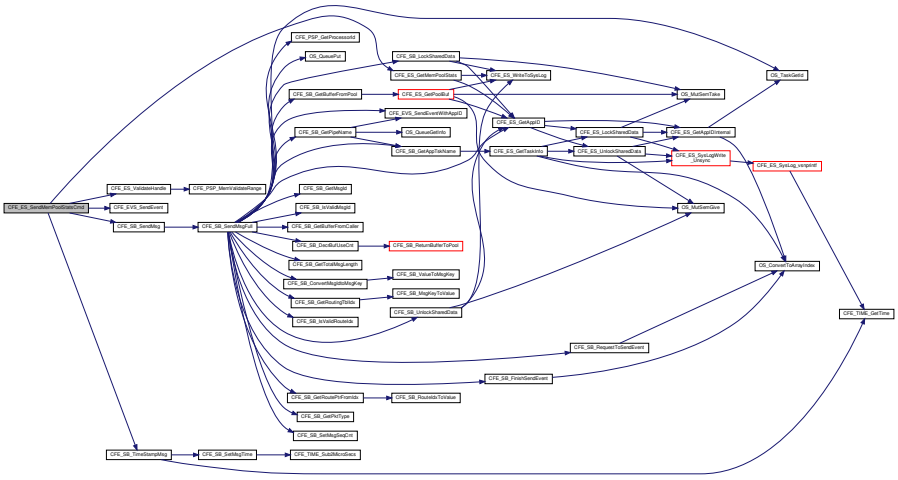
```
int32 CFE_ES_SendMemPoolStatsCmd (
            const CFE_ES_SendMemPoolStats_t * data )
```

Definition at line 1812 of file cfe_es_task.c.

References CFE_ES_GetMemPoolStats(), CFE_ES_INVALID_POOL_HANDLE_ERR_EID, CFE_ES_TLM_POOL_STATS_INFO_EID, CFE_ES_ValidateHandle(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GET_MEMADDR, CFE_SB_SendMsg(), CFE_SB_SET_MEMADDR, CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_TaskData_t::MemStatsPacket, CFE_ES_SendMemPoolStats_t::Payload, CFE_ES_MemStatsTlm_t::Payload, CFE_ES_SendMemPoolStatsCmd_Payload_t::PoolHandle, CFE_ES_PoolStatsTlm_Payload_t::PoolHandle, and CFE_ES_PoolStatsTlm_Payload_t::PoolStats.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.19 CFE_ES_SetMaxPRCountCmd()

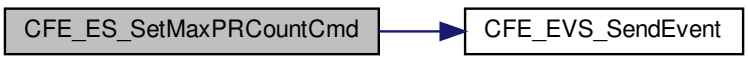
```
int32 CFE_ES_SetMaxPRCountCmd (
            const CFE_ES_SetMaxPRCount_t * data )
```

Definition at line 1725 of file cfe_es_task.c.

References CFE_ES_ResetDataPtr, CFE_ES_SET_MAX_PR_COUNT_EID, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_SetMaxPRCount↩Cmd_Payload_t::MaxPRCount, CFE_ES_ResetVariables_t::MaxProcessorResetCount, CFE_ES_SetMaxPRCount_↩t::Payload, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.20 CFE_ES_ShellCmd()

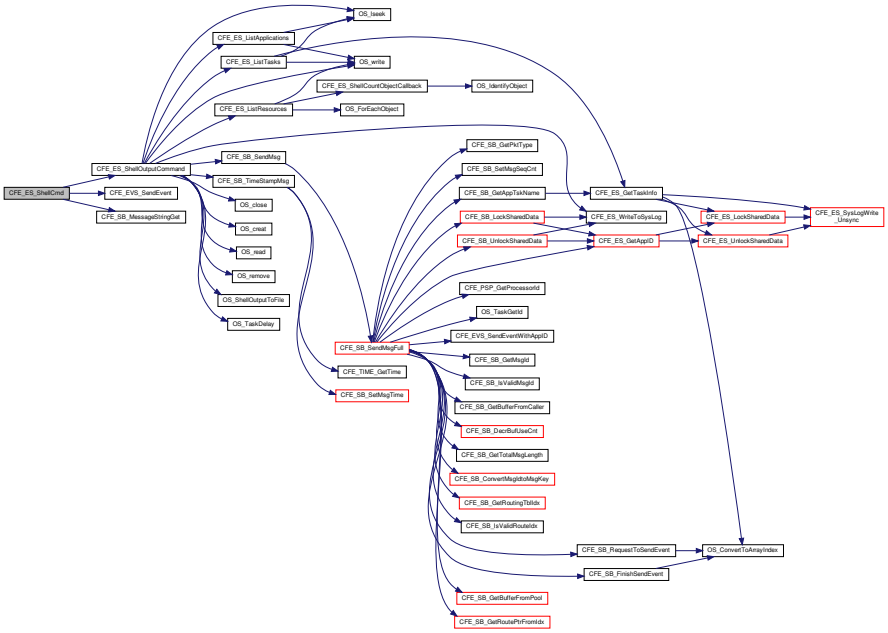
```
int32 CFE_ES_ShellCmd (
            const CFE_ES_Shell_t * data )
```

Definition at line 831 of file cfe_es_task.c.

References CFE_ES_SHELL_ERR_EID, CFE_ES_SHELL_INF_EID, CFE_ES_ShellOutputCommand(), CFE_EVS_↩
EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_PLATFORM_ES_MAX_↩
SHELL_CMD, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_ShellCmd_Payload_t::CmdString, CFE_E↩
S_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, OS_MAX_PATH_LEN, CF↩
E_ES_ShellCmd_Payload_t::OutputFilename, and CFE_ES_Shell_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.21 CFE_ES_StartAppCmd()

```
int32 CFE_ES_StartAppCmd (
            const CFE_ES_StartApp_t * data )
```

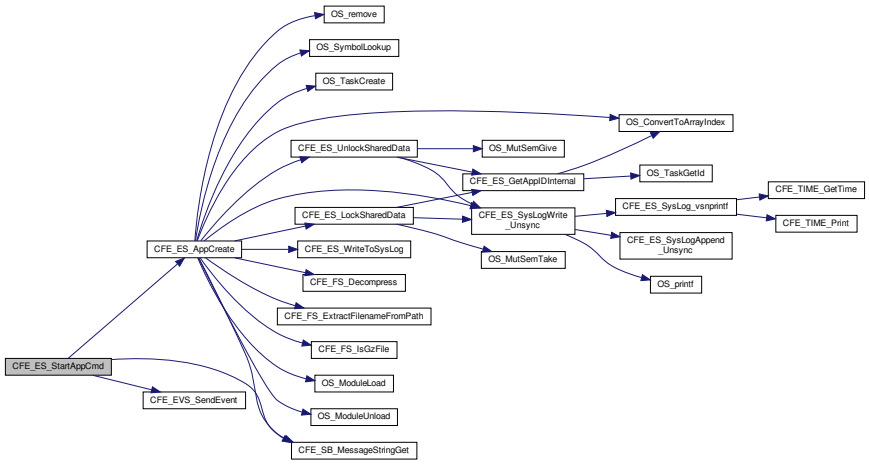
Definition at line 877 of file cfe_es_task.c.

References CFE_ES_StartAppCmd_Payload_t::AppEntryPoint, CFE_ES_StartAppCmd_Payload_t::AppFileName, C↩
FE_ES_StartAppCmd_Payload_t::Application, CFE_ES_AppCreate(), CFE_ES_ExceptionAction_PROC_RESTART,

CFE_ES_ExceptionAction_RESTART_APP, CFE_ES_START_ERR_EID, CFE_ES_START_EXC_ACTION_ERR_↩ EID, CFE_ES_START_INF_EID, CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID, CFE_ES_START_INVA↩ LID_FILENAME_ERR_EID, CFE_ES_START_NULL_APP_NAME_ERR_EID, CFE_ES_START_PRIORITY_ERR↩ _EID, CFE_ES_START_STACK_ERR_EID, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_PLATFORM_ES_DEFAULT_STACK_SIZE, CFE_SB_MessageStringGet(), CFE_S↩ UCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_Start↩ AppCmd_Payload_t::ExceptionAction, NULL, OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_MAX_PRIORITY, CFE_ES_StartApp_t::Payload, CFE_ES_StartAppCmd_Payload_t::Priority, and CFE_ES_StartAppCmd_Payload_t::↩ StackSize.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.22 CFE_ES_StopAppCmd()

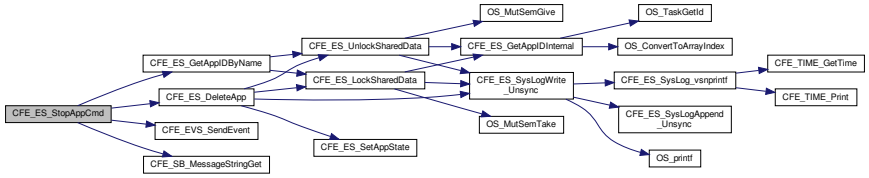
```
int32 CFE_ES_StopAppCmd (
            const CFE_ES_StopApp_t * data )
```

Definition at line 985 of file cfe_es_task.c.

References CFE_ES_AppNameCmd_Payload_t::Application, CFE_ES_DeleteApp(), CFE_ES_GetAppIDByName(), CFE_ES_STOP_DBG_EID, CFE_ES_STOP_ERR1_EID, CFE_ES_STOP_ERR2_EID, CFE_EVS_EventType_D↩ EBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, OS_MAX_API_NA↩ ME, and CFE_ES_AppNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.23 CFE_ES_TaskInit()

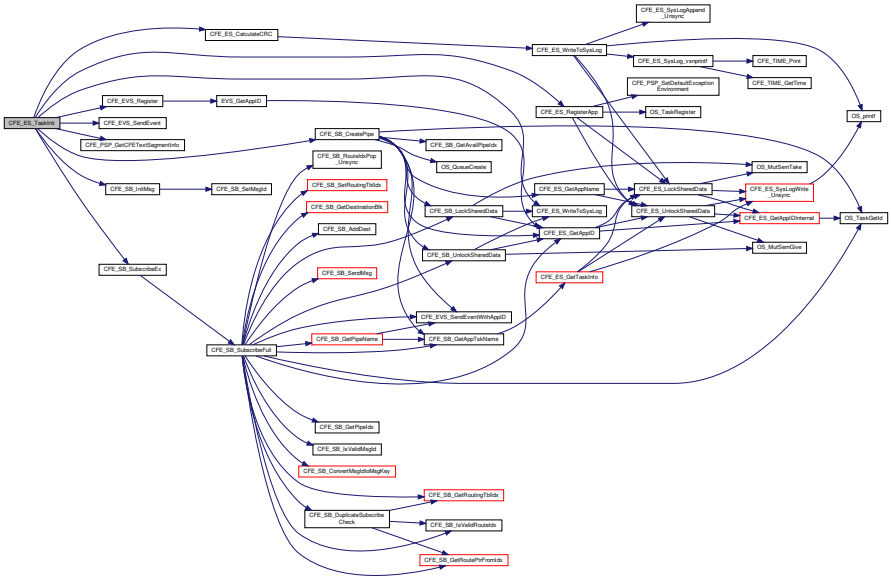
```
int32 CFE_ES_TaskInit (
            void  )
```

Definition at line 203 of file cfe_es_task.c.

References CFE_ES_APP_TLM_MID, CFE_ES_BUILD_INF_EID, CFE_ES_CalculateCRC(), CFE_ES_CMD_M↩ID, CFE_ES_HK_TLM_MID, CFE_ES_INIT_INF_EID, CFE_ES_INITSTATS_INF_EID, CFE_ES_MEMSTATS_TL↩M_MID, CFE_ES_RegisterApp(), CFE_ES_ResetDataPtr, CFE_ES_SEND_HK_MID, CFE_ES_SHELL_TLM_MID, CFE_ES_VERSION_INF_EID, CFE_ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType↩_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_MAJOR_VERSION, CFE_MINOR_VERS↩ION, CFE_MISSION_ES_DEFAULT_CRC, CFE_MISSION_EVS_MAX_MESSAGE_LENGTH, CFE_MISSION_REV, CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE, CFE_PSP_GetCFETextSegmentInfo(), CFE_PSP_MAJOR_V↩ERSION, CFE_PSP_MINOR_VERSION, CFE_PSP_MISSION_REV, CFE_PSP_REVISION, CFE_PSP_SUCCESS, CFE_REVISION, CFE_SB_CreatePipe(), CFE_SB_Default_Qos, CFE_SB_InitMsg(), CFE_SB_SubscribeEx(), CFE↩_SUCCESS, CFE_ES_HousekeepingTlm_Payload_t::CFECoreChecksum, CFE_ES_HousekeepingTlm_Payload_t::↩CFEMajorVersion, CFE_ES_HousekeepingTlm_Payload_t::CFEMinorVersion, CFE_ES_HousekeepingTlm_Payload↩_t::CFEMissionRevision, CFE_ES_HousekeepingTlm_Payload_t::CFERevision, CFE_ES_TaskData_t::CmdPipe, C↩FE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_TaskData_t::Hk↩Packet, CFE_ES_TaskData_t::LimitCmd, CFE_ES_TaskData_t::LimitHK, CFE_ES_TaskData_t::MemStatsPacket, N↩ULL, CFE_ES_TaskData_t::OneAppPacket, OS_MAJOR_VERSION, OS_MINOR_VERSION, OS_MISSION_REV, OS_REVISION, CFE_ES_HousekeepingTlm_Payload_t::OSALMajorVersion, CFE_ES_HousekeepingTlm_Payload↩_t::OSALMinorVersion, CFE_ES_HousekeepingTlm_Payload_t::OSALMissionRevision, CFE_ES_HousekeepingTlm↩_Payload_t::OSALRevision, CFE_ES_HousekeepingTlm_t::Payload, CFE_ES_TaskData_t::PipeDepth, CFE_ES_↩TaskData_t::PipeName, CFE_ES_TaskData_t::ShellPacket, and CFE_ES_ResetData_t::SystemLogMode.

Referenced by CFE_ES_TaskMain().

Here is the call graph for this function:



13.33.2.24 CFE_ES_TaskMain()

```
void CFE_ES_TaskMain (
             void  )
```

Description

This is the entry point to the cFE ES Core Application.

Assumptions, External Events, and Notes:

None

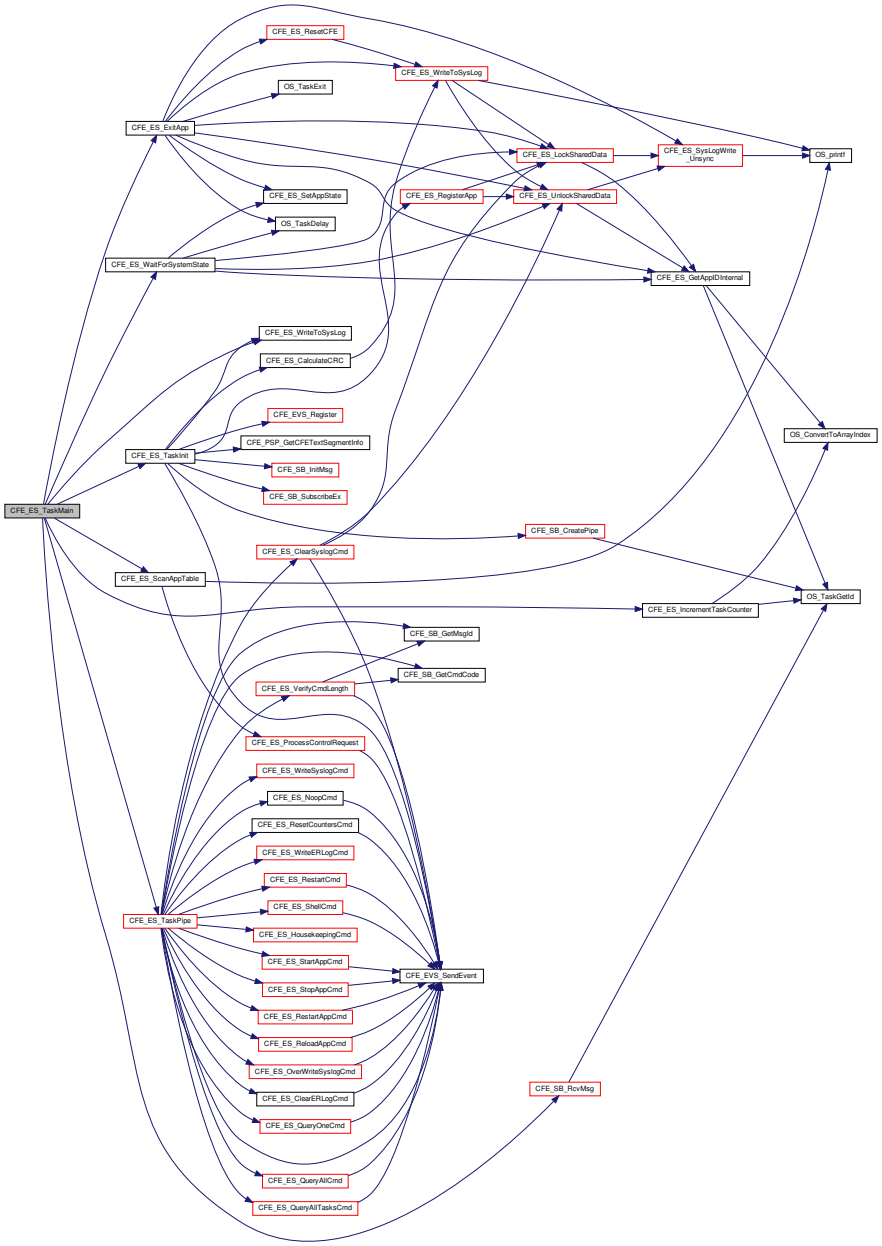
Return values

None	

Definition at line 80 of file cfe_es_task.c.

References CFE_ES_ExitApp(), CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunStatus_APP_RUN, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_A↩PP_RUNTIME_ERROR, CFE_ES_ScanAppTable(), CFE_ES_SystemState_CORE_READY, CFE_ES_TaskInit(), CF↩E_ES_TaskPipe(), CFE_ES_WaitForSystemState(), CFE_ES_WriteToSysLog(), CFE_MISSION_ES_MAIN_PERF_ID, CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_PLATFORM_ES_APP_SCAN_RATE, CFE_SB_RcvMsg(), CFE_SB_TIME_OUT, CFE_SUCCESS, CFE_ES_TaskData_t::CmdPipe, and CFE_ES_TaskData_t::MsgPtr.

Here is the call graph for this function:



13.33.2.25 CFE_ES_TaskPipe()

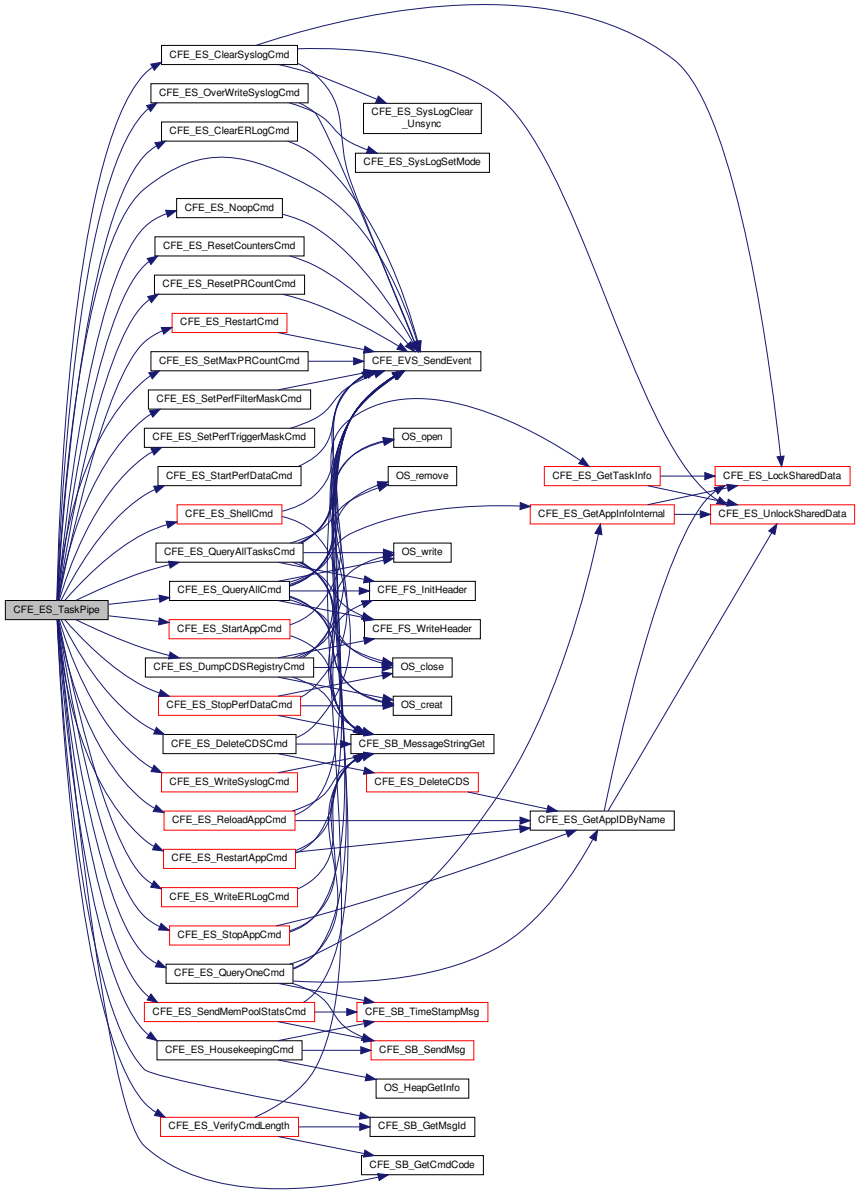
```
void CFE_ES_TaskPipe (
            CFE_SB_MsgPtr_t Msg )
```

Definition at line 413 of file cfe_es_task.c.

References CFE_ES_CC1_ERR_EID, CFE_ES_CLEAR_ER_LOG_CC, CFE_ES_CLEAR_SYSLOG_CC, CFE_ES_ClearERLogCmd(), CFE_ES_ClearSyslogCmd(), CFE_ES_CMD_MID, CFE_ES_DELETE_CDS_CC, CFE_ES_DeleteCDSCmd(), CFE_ES_DUMP_CDS_REGISTRY_CC, CFE_ES_DumpCDSRegistryCmd(), CFE_ES_HousekeepingCmd(), CFE_ES_MID_ERR_EID, CFE_ES_NOOP_CC, CFE_ES_NoopCmd(), CFE_ES_OVER_WRITE_SYSLOG_CC, CFE_ES_OverWriteSyslogCmd(), CFE_ES_QUERY_ALL_CC, CFE_ES_QUERY_ALL_TASKS_CC, CFE_ES_QUERY_ONE_CC, CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_RELOAD_APP_CC, CFE_ES_ReloadAppCmd(), CFE_ES_RESET_COUNTERS_CC, CFE_ES_RESET_PR_COUNT_CC, CFE_ES_ResetCountersCmd(), CFE_ES_ResetPRCountCmd(), CFE_ES_RESTART_APP_CC, CFE_ES_RESTART_CC, CFE_ES_RestartAppCmd(), CFE_ES_RestartCmd(), CFE_ES_SEND_HK_MID, CFE_ES_SEND_MEM_POOL_STATS_CC, CFE_ES_SendMemPoolStatsCmd(), CFE_ES_SET_MAX_PR_COUNT_CC, CFE_ES_SET_PERF_FILTER_MASK_CC, CFE_ES_SET_PERF_TRIGGER_MASK_CC, CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_SHELL_CC, CFE_ES_ShellCmd(), CFE_ES_START_APP_CC, CFE_ES_START_PERF_DATA_CC, CFE_ES_StartAppCmd(), CFE_ES_StartPerfDataCmd(), CFE_ES_STOP_APP_CC, CFE_ES_STOP_PERF_DATA_CC, CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_VerifyCmdLength(), CFE_ES_WRITE_ER_LOG_CC, CFE_ES_WRITE_SYSLOG_CC, CFE_ES_WriteERLogCmd(), CFE_ES_WriteSyslogCmd(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(), and CFE_ES_TaskData_t::CommandErrorCounter.

Referenced by CFE_ES_TaskMain().

Here is the call graph for this function:



13.33.2.26 CFE_ES_VerifyCmdLength()

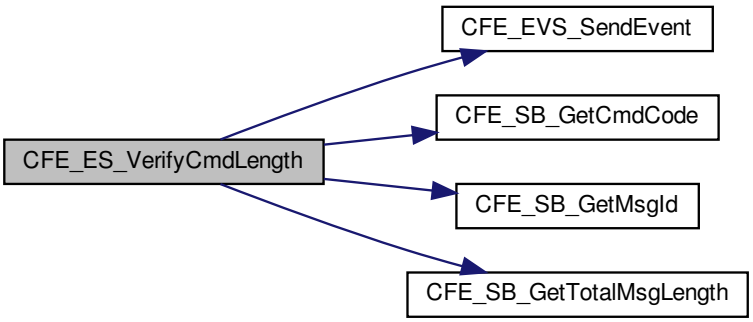
```
bool CFE_ES_VerifyCmdLength (
            CFE_SB_MsgPtr_t Msg,
            uint16 ExpectedLength )
```

Definition at line 1669 of file cfe_es_task.c.

References CFE_ES_LEN_ERR_EID, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GetCmd↩ Code(), CFE_SB_GetMsgId(), CFE_SB_GetTotalMsgLength(), and CFE_ES_TaskData_t::CommandErrorCounter.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.27 CFE_ES_WriteERLogCmd()

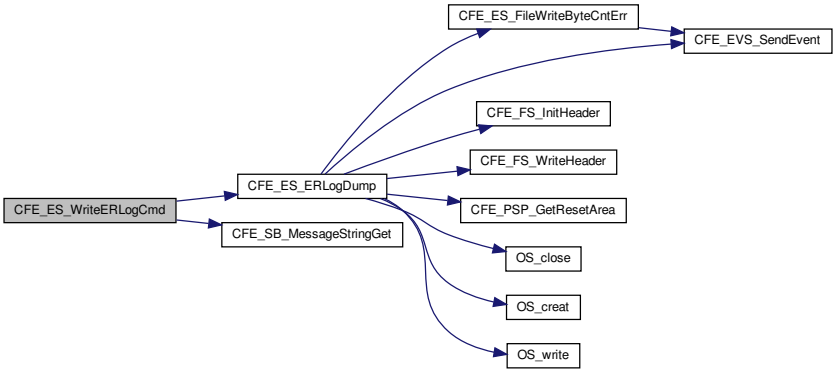
```
int32 CFE_ES_WriteERLogCmd (
            const CFE_ES_WriteERLog_t * data )
```

Definition at line 1571 of file cfe_es_task.c.

References CFE_ES_ERLogDump(), CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE, CFE_SB_MessageString↩ Get(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, C↩ FE_ES_FileNameCmd_Payload_t::FileName, OS_MAX_PATH_LEN, and CFE_ES_FileNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.2.28 CFE_ES_WriteSyslogCmd()

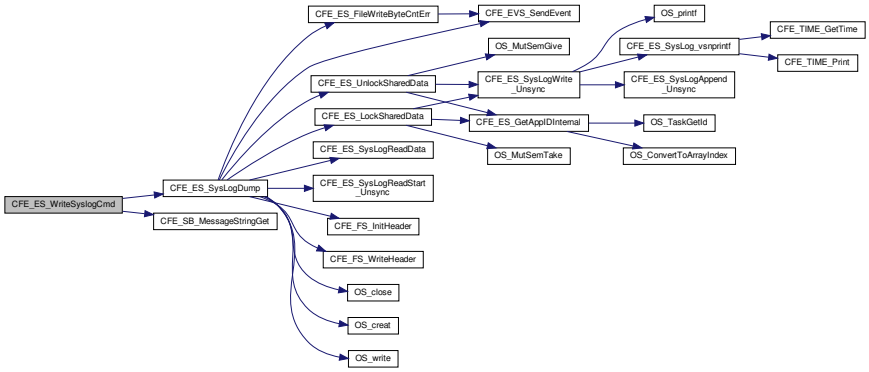
```
int32 CFE_ES_WriteSyslogCmd (
            const CFE_ES_WriteSyslog_t * data )
```

Definition at line 1506 of file cfe_es_task.c.

References CFE_ES_SysLogDump(), CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_MAX_PATH_LEN, and CFE_ES_FileNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.33.3 Variable Documentation

13.33.3.1 CFE_ES_TaskData

```
CFE_ES_TaskData_t CFE_ES_TaskData
```

Definition at line 72 of file cfe_es_task.c.

Referenced by CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_ShellOutputCommand(), CFE_ES_StartPerfDataCmd(), CFE_ES_StopPerfDataCmd(), and CFE_ES_SysLogDump().

13.34 cfe/fsw/cfe-core/src/es/cfe_es_task.h File Reference

```
#include "cfe.h"
#include "cfe_es.h"
#include "cfe_es_apps.h"
#include "cfe_es_events.h"
#include "cfe_es_msg.h"
```

Data Structures

- struct CFE_ES_TaskData_t

Macros

- #define CFE_ES_SYS_LOG_DESC "ES system log data file"
- #define CFE_ES_TASK_LOG_DESC "ES Task Info file"
- #define CFE_ES_APP_LOG_DESC "ES Application Info file"
- #define CFE_ES_ER_LOG_DESC "ES ERlog data file"
- #define CFE_ES_PERF_LOG_DESC "ES Performance data file"

Functions

- void CFE_ES_TaskMain (void)

  Entry Point for cFE Core Application.
- int32 CFE_ES_TaskInit (void)
- void CFE_ES_TaskPipe (CFE_SB_MsgPtr_t Msg)
- int32 CFE_ES_HousekeepingCmd (const CCSDS_CommandPacket_t *data)
- int32 CFE_ES_NoopCmd (const CFE_ES_Noop_t *Cmd)
- int32 CFE_ES_ResetCountersCmd (const CFE_ES_ResetCounters_t *data)
- int32 CFE_ES_RestartCmd (const CFE_ES_Restart_t *data)
- int32 CFE_ES_ShellCmd (const CFE_ES_Shell_t *data)
- int32 CFE_ES_StartAppCmd (const CFE_ES_StartApp_t *data)
- int32 CFE_ES_StopAppCmd (const CFE_ES_StopApp_t *data)
- int32 CFE_ES_RestartAppCmd (const CFE_ES_RestartApp_t *data)
- int32 CFE_ES_ReloadAppCmd (const CFE_ES_ReloadApp_t *data)
- int32 CFE_ES_QueryOneCmd (const CFE_ES_QueryOne_t *data)
- int32 CFE_ES_QueryAllCmd (const CFE_ES_QueryAll_t *data)
- int32 CFE_ES_QueryAllTasksCmd (const CFE_ES_QueryAllTasks_t *data)
- int32 CFE_ES_ClearSyslogCmd (const CFE_ES_ClearSyslog_t *data)
- int32 CFE_ES_OverWriteSyslogCmd (const CFE_ES_OverWriteSyslog_t *data)
- int32 CFE_ES_WriteSyslogCmd (const CFE_ES_WriteSyslog_t *data)
- int32 CFE_ES_ClearERLogCmd (const CFE_ES_ClearERLog_t *data)
- int32 CFE_ES_WriteERLogCmd (const CFE_ES_WriteERLog_t *data)
- int32 CFE_ES_ResetPRCountCmd (const CFE_ES_ResetPRCount_t *data)
- int32 CFE_ES_SetMaxPRCountCmd (const CFE_ES_SetMaxPRCount_t *data)
- int32 CFE_ES_DeleteCDSCmd (const CFE_ES_DeleteCDS_t *data)
- int32 CFE_ES_StartPerfDataCmd (const CFE_ES_StartPerfData_t *data)
- int32 CFE_ES_StopPerfDataCmd (const CFE_ES_StopPerfData_t *data)
- int32 CFE_ES_SetPerfFilterMaskCmd (const CFE_ES_SetPerfFilterMask_t *data)
- int32 CFE_ES_SetPerfTriggerMaskCmd (const CFE_ES_SetPerfTriggerMask_t *data)
- int32 CFE_ES_SendMemPoolStatsCmd (const CFE_ES_SendMemPoolStats_t *data)
- int32 CFE_ES_DumpCDSRegistryCmd (const CFE_ES_DumpCDSRegistry_t *data)
- bool CFE_ES_ValidateHandle (CFE_ES_MemHandle_t Handle)
- bool CFE_ES_VerifyCmdLength (CFE_SB_MsgPtr_t msg, uint16 ExpectedLength)
- void CFE_ES_FileWriteByteCntErr (const char *Filename, uint32 Requested, uint32 Actual)

Variables

- CFE_ES_TaskData_t CFE_ES_TaskData

13.34.1 Macro Definition Documentation

13.34.1.1 CFE_ES_APP_LOG_DESC

```
#define CFE_ES_APP_LOG_DESC "ES Application Info file"
```

Definition at line 54 of file cfe_es_task.h.

Referenced by CFE_ES_QueryAllCmd().

13.34.1.2 CFE_ES_ER_LOG_DESC

```
#define CFE_ES_ER_LOG_DESC "ES ERlog data file"
```

Definition at line 55 of file cfe_es_task.h.

Referenced by CFE_ES_ERLogDump().

13.34.1.3 CFE_ES_PERF_LOG_DESC

```
#define CFE_ES_PERF_LOG_DESC "ES Performance data file"
```

Definition at line 56 of file cfe_es_task.h.

Referenced by CFE_ES_PerfLogDump().

13.34.1.4 CFE_ES_SYS_LOG_DESC

```
#define CFE_ES_SYS_LOG_DESC "ES system log data file"
```

Definition at line 52 of file cfe_es_task.h.

Referenced by CFE_ES_SysLogDump().

13.34.1.5 CFE_ES_TASK_LOG_DESC

```
#define CFE_ES_TASK_LOG_DESC "ES Task Info file"
```

Definition at line 53 of file cfe_es_task.h.

Referenced by CFE_ES_QueryAllTasksCmd().

13.34.2 Function Documentation

13.34.2.1 CFE_ES_ClearERLogCmd()

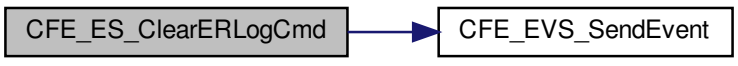
```
int32 CFE_ES_ClearERLogCmd (
            const CFE_ES_ClearERLog_t * data )
```

Definition at line 1536 of file cfe_es_task.c.

References CFE_ES_ERLOG1_INF_EID, CFE_ES_ResetDataPtr, CFE_EVS_EventType_INFORMATION, CFE_EV↩S_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_ResetData_t::ERLog, CFE_E↩S_ResetData_t::ERLogEntries, and CFE_ES_ResetData_t::ERLogIndex.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.2 CFE_ES_ClearSyslogCmd()

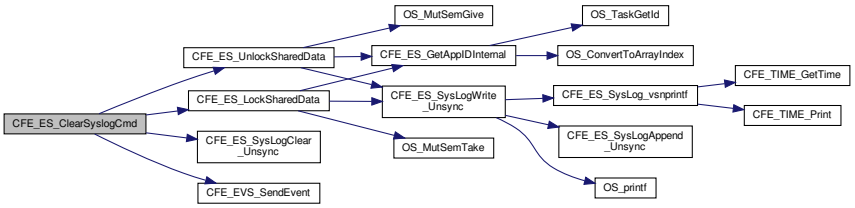
```
int32 CFE_ES_ClearSyslogCmd (
            const CFE_ES_ClearSyslog_t * data )
```

Definition at line 1446 of file cfe_es_task.c.

References CFE_ES_LockSharedData(), CFE_ES_SYSLOG1_INF_EID, CFE_ES_SysLogClear_Unsync(), CFE_ES_UnlockSharedData(), CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, and CFE_ES_TaskData_t::CommandCounter.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.3 CFE_ES_DeleteCDSCmd()

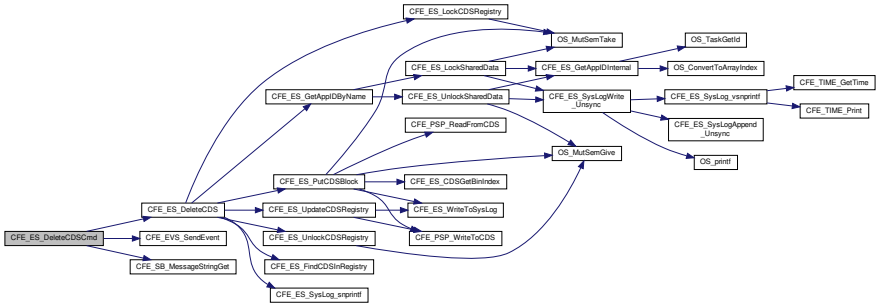
```
int32 CFE_ES_DeleteCDSCmd (
            const CFE_ES_DeleteCDS_t * data )
```

Definition at line 1751 of file cfe_es_task.c.

References CFE_ES_DeleteCDSCmd_Payload_t::CdsName, CFE_ES_CDS_DELETE_ERR_EID, CFE_ES_CDS_DELETE_TBL_ERR_EID, CFE_ES_CDS_DELETED_INFO_EID, CFE_ES_CDS_MAX_FULL_NAME_LEN, CFE_ES_CDS_NAME_ERR_EID, CFE_ES_CDS_NOT_FOUND_ERR, CFE_ES_CDS_OWNER_ACTIVE_EID, CFE_ES_CDS_OWNER_ACTIVE_ERR, CFE_ES_CDS_WRONG_TYPE_ERR, CFE_ES_DeleteCDS(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, and CFE_ES_DeleteCDS_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.4 CFE_ES_DumpCDSRegistryCmd()

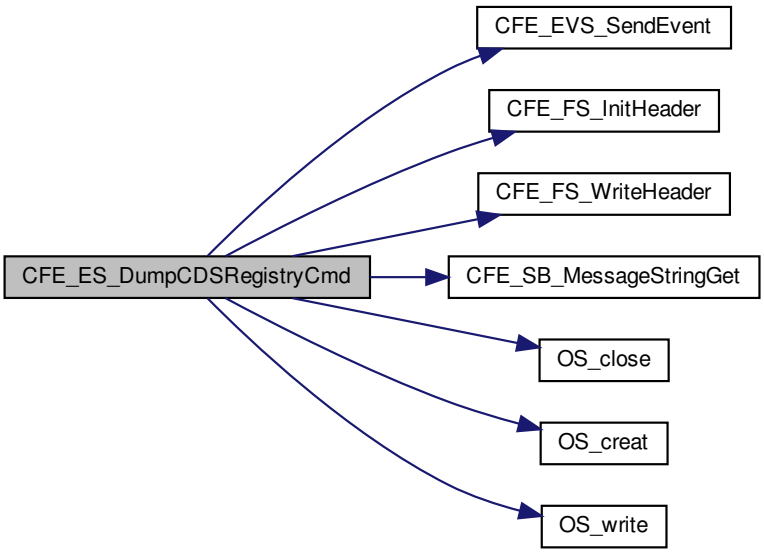
```
int32 CFE_ES_DumpCDSRegistryCmd (
            const CFE_ES_DumpCDSRegistry_t * data )
```

Definition at line 1859 of file cfe_es_task.c.

References CFE_ES_CDSRegDumpRec_t::ByteAlignSpare1, CFE_ES_Global_t::CDSVars, CFE_ES_CDS_DUMP_ERR_EID, CFE_ES_CDS_REG_DUMP_INF_EID, CFE_ES_CREATING_CDS_DUMP_ERR_EID, CFE_ES_Global, CFE_ES_WRITE_CFE_HDR_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_CDS_REG, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES, CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_DumpCDSRegistryCmd_Payload_t::DumpFilename, CFE_ES_CDSRegDumpRec_t::Handle, CFE_ES_CDS_RegRec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, CFE_ES_CDSRegDumpRec_t::Name, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_MAX_PATH_LEN, OS_write(), OS_WRITE_ONLY, CFE_ES_DumpCDSRegistry_t::Payload, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec_t::Size, CFE_ES_CDSRegDumpRec_t::Size, CFE_ES_CDS_RegRec_t::Table, CFE_ES_CDSRegDumpRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.5 CFE_ES_FileWriteByteCntErr()

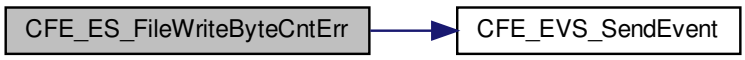
```
void CFE_ES_FileWriteByteCntErr (
            const char * Filename,
            uint32 Requested,
            uint32 Actual )
```

Definition at line 1978 of file cfe_es_task.c.

References CFE_ES_FILEWRITE_ERR_EID, CFE_EVS_EventType_ERROR, and CFE_EVS_SendEvent().

Referenced by CFE_ES_ERLogDump(), CFE_ES_PerfLogDump(), and CFE_ES_SysLogDump().

Here is the call graph for this function:



13.34.2.6 CFE_ES_HousekeepingCmd()

```
int32 CFE_ES_HousekeepingCmd (
            const CCSDS_CommandPacket_t * data )
```

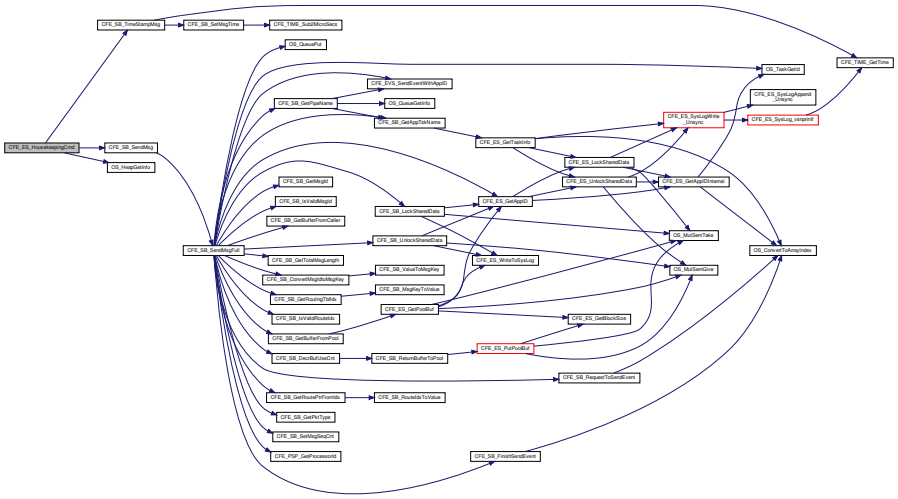
Definition at line 638 of file cfe_es_task.c.

References CFE_ES_ResetVariables_t::BootSource, CFE_ES_HousekeepingTlm_Payload_t::BootSource, CFE_E↩S_Global, CFE_ES_PERF_FILTERMASK_EXT_SIZE, CFE_ES_PERF_FILTERMASK_INT_SIZE, CFE_ES_PERF↩_TRIGGERMASK_EXT_SIZE, CFE_ES_PERF_TRIGGERMASK_INT_SIZE, CFE_ES_PerfLogDumpStatus, CFE_↩ES_ResetDataPtr, CFE_PLATFORM_ES_SYSTEM_LOG_SIZE, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_HousekeepingTlm_Payload_t::Command↩Counter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_HousekeepingTlm_Payload_t::CommandError↩Counter, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfMetaData_t::↩DataStart, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_ResetData_t::ERLogEntries, CFE_ES_Housekeeping↩Tlm_Payload_t::ERLogEntries, CFE_ES_ResetData_t::ERLogIndex, CFE_ES_HousekeepingTlm_Payload_t::ERLog↩Index, CFE_ES_PerfMetaData_t::FilterMask, OS_heap_prop_t::free_blocks, OS_heap_prop_t::free_bytes, CFE_E↩S_HousekeepingTlm_Payload_t::HeapBlocksFree, CFE_ES_HousekeepingTlm_Payload_t::HeapBytesFree, CFE_↩ES_HousekeepingTlm_Payload_t::HeapMaxBlockSize, CFE_ES_TaskData_t::HkPacket, OS_heap_prop_t::largest↩_free_block, CFE_ES_ResetVariables_t::MaxProcessorResetCount, CFE_ES_HousekeepingTlm_Payload_t::Max↩ProcessorResets, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, OS_HeapGetInfo(), OS_SUC↩CESS, CFE_ES_HousekeepingTlm_t::Payload, CFE_ES_ResetData_t::Perf, CFE_ES_HousekeepingTlm_Payload↩_t::PerfDataCount, CFE_ES_HousekeepingTlm_Payload_t::PerfDataEnd, CFE_ES_HousekeepingTlm_Payload_↩t::PerfDataStart, CFE_ES_HousekeepingTlm_Payload_t::PerfDataToWrite, CFE_ES_HousekeepingTlm_Payload↩_t::PerfFilterMask, CFE_ES_HousekeepingTlm_Payload_t::PerfMode, CFE_ES_HousekeepingTlm_Payload_t::↩PerfState, CFE_ES_HousekeepingTlm_Payload_t::PerfTriggerCount, CFE_ES_HousekeepingTlm_Payload_t::Perf↩

TriggerMask, CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_HousekeepingTlm_Payload_t::Processor↩ Resets, CFE_ES_Global_t::RegisteredCoreApps, CFE_ES_HousekeepingTlm_Payload_t::RegisteredCoreApps, C↩ FE_ES_Global_t::RegisteredExternalApps, CFE_ES_HousekeepingTlm_Payload_t::RegisteredExternalApps, C↩ FE_ES_Global_t::RegisteredLibs, CFE_ES_HousekeepingTlm_Payload_t::RegisteredLibs, CFE_ES_Global_t::↩ RegisteredTasks, CFE_ES_HousekeepingTlm_Payload_t::RegisteredTasks, CFE_ES_ResetVariables_t::Reset↩ Subtype, CFE_ES_HousekeepingTlm_Payload_t::ResetSubtype, CFE_ES_ResetVariables_t::ResetType, CFE_↩ ES_HousekeepingTlm_Payload_t::ResetType, CFE_ES_ResetData_t::ResetVars, CFE_ES_PerfMetaData_t::State, CFE_ES_HousekeepingTlm_Payload_t::SysLogBytesUsed, CFE_ES_HousekeepingTlm_Payload_t::SysLogEntries, CFE_ES_HousekeepingTlm_Payload_t::SysLogMode, CFE_ES_HousekeepingTlm_Payload_t::SysLogSize, CFE_↩ ES_ResetData_t::SystemLogEndIdx, CFE_ES_ResetData_t::SystemLogEntryNum, CFE_ES_ResetData_t::System↩ LogMode, CFE_ES_PerfMetaData_t::TriggerCount, and CFE_ES_PerfMetaData_t::TriggerMask.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.7 CFE_ES_NoopCmd()

```
int32 CFE_ES_NoopCmd (
            const CFE_ES_Noop_t * Cmd )
```

Definition at line 748 of file cfe_es_task.c.

References CFE_ES_BUILD_INF_EID, CFE_ES_NOOP_INF_EID, CFE_EVS_EventType_INFORMATION, CFE_E↩ VS_SendEvent(), CFE_MAJOR_VERSION, CFE_MINOR_VERSION, CFE_MISSION_REV, CFE_PSP_MAJOR_VE↩ RSION, CFE_PSP_MINOR_VERSION, CFE_PSP_MISSION_REV, CFE_PSP_REVISION, CFE_REVISION, CFE_↩ SUCCESS, CFE_ES_TaskData_t::CommandCounter, OS_MAJOR_VERSION, OS_MINOR_VERSION, OS_MISSIO↩ N_REV, and OS_REVISION.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:

13.34.2.8 CFE_ES_OverWriteSyslogCmd()

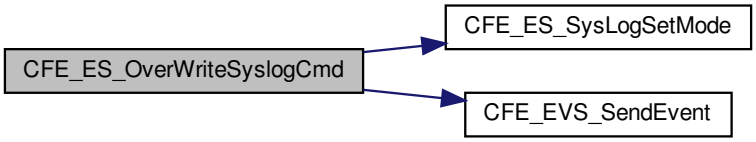
```
int32 CFE_ES_OverWriteSyslogCmd (
            const CFE_ES_OverWriteSyslog_t * data )
```

Definition at line 1472 of file cfe_es_task.c.

References CFE_ES_ERR_SYSLOGMODE_EID, CFE_ES_SYSLOGMODE_EID, CFE_ES_SysLogSetMode(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_OverWriteSysLogCmd_Payload_t::Mode, and CFE_ES_OverWriteSyslog_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.9 CFE_ES_QueryAllCmd()

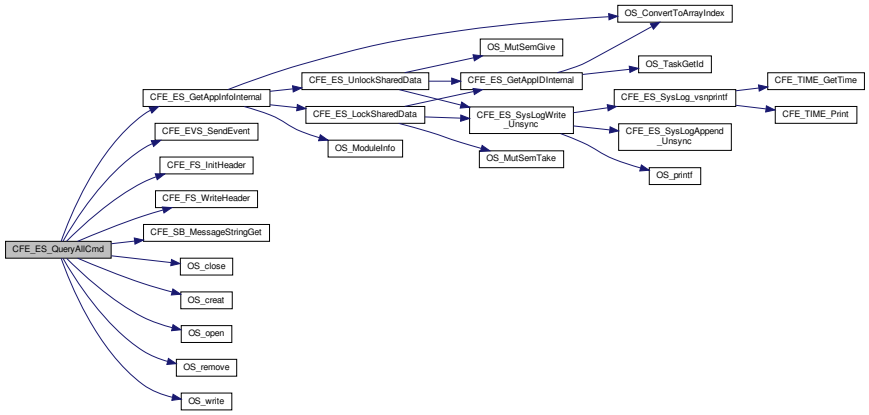
```
int32 CFE_ES_QueryAllCmd (
            const CFE_ES_QueryAll_t * data )
```

Definition at line 1194 of file cfe_es_task.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ALL_APPS_EID, CFE_ES_APP_LOG_DESC, CFE_ES_AppState_UNDEFINED, CFE_ES_GetAppInfoInternal(), CFE_ES_Global, CFE_ES_OSCREATE_ERR_EID, CFE_ES_TASKWR_ERR_EID, CFE_ES_WRHDR_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_QUERYALL, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_close(), OS_creat(), OS_MAX_PATH_LEN, OS_open(), OS_READ_ONLY, OS_remove(), OS_write(), OS_WRITE_ONLY, and CFE_ES_FileNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.10 CFE_ES_QueryAllTasksCmd()

```
int32 CFE_ES_QueryAllTasksCmd (
            const CFE_ES_QueryAllTasks_t * data )
```

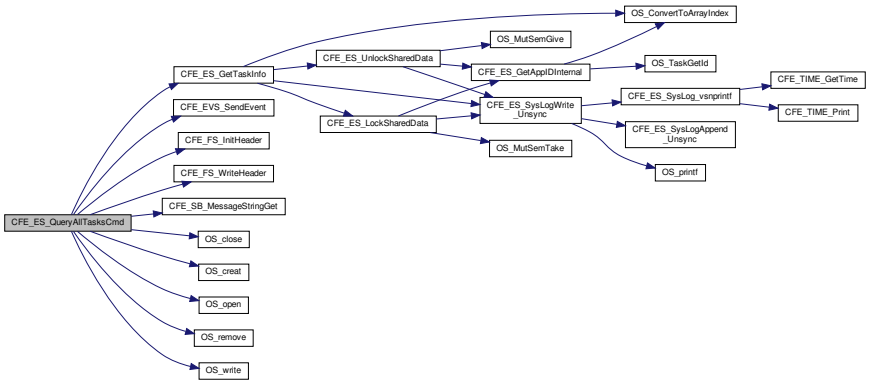
Definition at line 1319 of file cfe_es_task.c.

References CFE_ES_GetTaskInfo(), CFE_ES_Global, CFE_ES_TASK_LOG_DESC, CFE_ES_TASKINFO_EID, CFE_ES_TASKINFO_OSCREATE_ERR_EID, CFE_ES_TASKINFO_WR_ERR_EID, CFE_ES_TASKINFO_WRHDR_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_ES_QUERYALLTASKS, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_close(), OS_creat(), OS_MAX_PATH_LEN, OS_MAX_TASKS, OS_open(), OS_READ_ONLY, OS_remove(), OS_write(), OS_WRITE_ONLY,

CFE_ES_FileNameCmd_t::Payload, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_TaskRecord_t::TaskId, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.11 CFE_ES_QueryOneCmd()

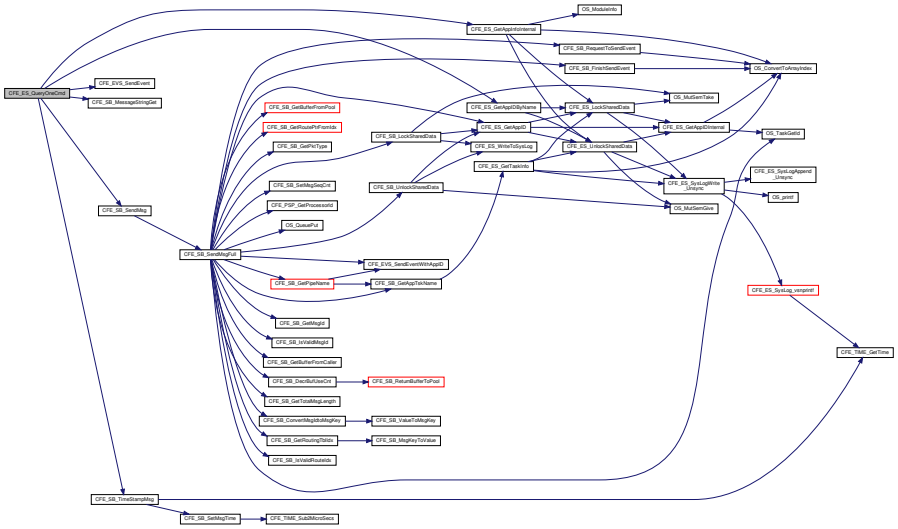
```
int32 CFE_ES_QueryOneCmd (
            const CFE_ES_QueryOne_t * data )
```

Definition at line 1140 of file cfe_es_task.c.

References CFE_ES_OneAppTlm_Payload_t::AppInfo, CFE_ES_AppNameCmd_Payload_t::Application, CFE_ES_GetAppIDByName(), CFE_ES_GetAppInfoInternal(), CFE_ES_ONE_APP_EID, CFE_ES_ONE_APPID_ERR_EID, CFE_ES_ONE_ERR_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, CFE_ES_TaskData_t::OneAppPacket, OS_MAX_API_NAME, CFE_ES_AppNameCmd_t::Payload, and CFE_ES_OneAppTlm_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.12 CFE_ES_ReloadAppCmd()

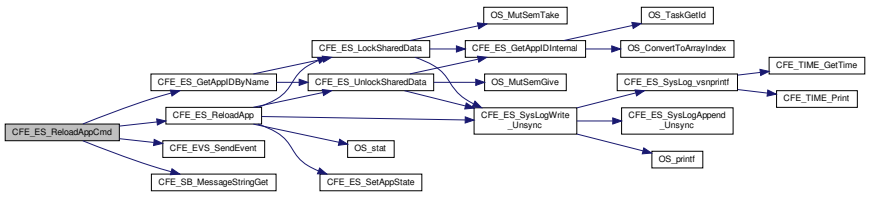
```
int32 CFE_ES_ReloadAppCmd (
            const CFE_ES_ReloadApp_t * data )
```

Definition at line 1087 of file cfe_es_task.c.

References CFE_ES_AppReloadCmd_Payload_t::AppFileName, CFE_ES_AppReloadCmd_Payload_t::Application, CFE_ES_GetAppIDByName(), CFE_ES_RELOAD_APP_DBG_EID, CFE_ES_RELOAD_APP_ERR1_EID, CFE_ES↩_RELOAD_APP_ERR2_EID, CFE_ES_ReloadApp(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, OS_MAX_API_NAME, OS_MAX_PATH_LEN, and CFE_ES_↩ReloadApp_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.13 CFE_ES_ResetCountersCmd()

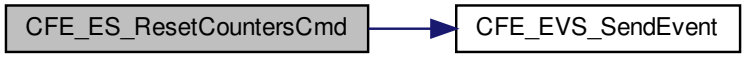
```
int32 CFE_ES_ResetCountersCmd (
            const CFE_ES_ResetCounters_t * data )
```

Definition at line 780 of file cfe_es_task.c.

References CFE_ES_RESET_INF_EID, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, and CFE_ES_TaskData_t::CommandErrorCounter.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.14 CFE_ES_ResetPRCountCmd()

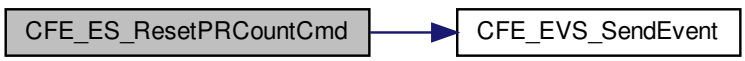
```
int32 CFE_ES_ResetPRCountCmd (
            const CFE_ES_ResetPRCount_t * data )
```

Definition at line 1701 of file cfe_es_task.c.

References CFE_ES_RESET_PR_COUNT_EID, CFE_ES_ResetDataPtr, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_ResetVariables_t::ProcessorResetCount, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.15 CFE_ES_RestartAppCmd()

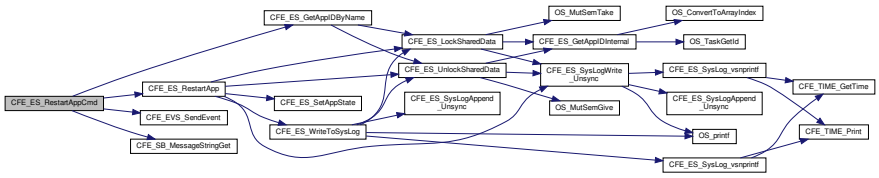
```
int32 CFE_ES_RestartAppCmd (
            const CFE_ES_RestartApp_t * data )
```

Definition at line 1038 of file cfe_es_task.c.

References CFE_ES_AppNameCmd_Payload_t::Application, CFE_ES_GetAppIDByName(), CFE_ES_RESTART_A↩PP_DBG_EID, CFE_ES_RESTART_APP_ERR1_EID, CFE_ES_RESTART_APP_ERR2_EID, CFE_ES_RestartApp(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageString↩Get(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, N↩ULL, OS_MAX_API_NAME, and CFE_ES_AppNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.16 CFE_ES_RestartCmd()

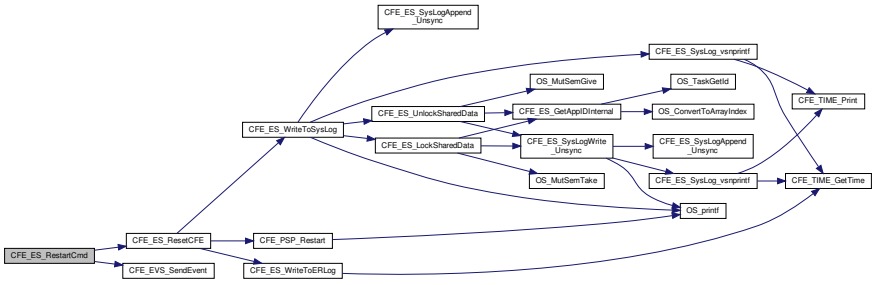
```
int32 CFE_ES_RestartCmd (
            const CFE_ES_Restart_t * data )
```

Definition at line 801 of file cfe_es_task.c.

References CFE_ES_BOOT_ERR_EID, CFE_ES_ResetCFE(), CFE_EVS_EventType_ERROR, CFE_EVS_Send↩Event(), CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_SUCCESS, CFE_ES_↩TaskData_t::CommandErrorCounter, CFE_ES_Restart_t::Payload, and CFE_ES_RestartCmd_Payload_t::RestartType.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.17 CFE_ES_SendMemPoolStatsCmd()

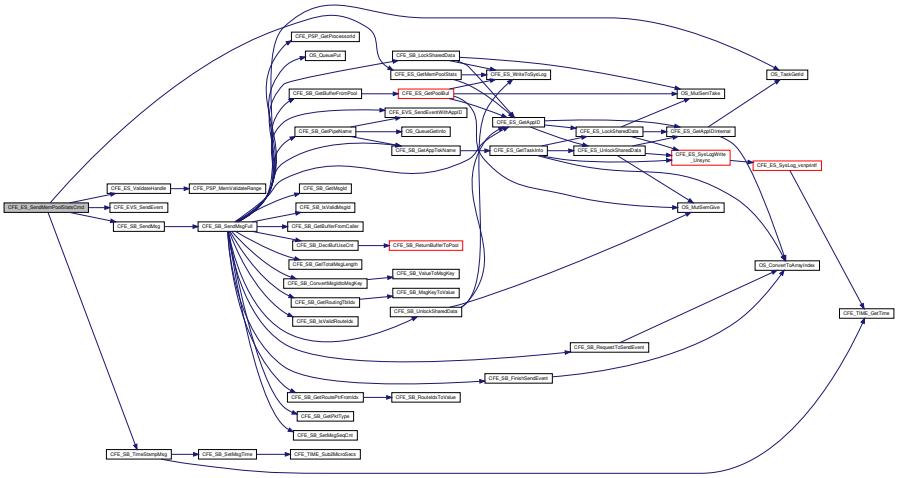
```
int32 CFE_ES_SendMemPoolStatsCmd (
            const CFE_ES_SendMemPoolStats_t * data )
```

Definition at line 1812 of file cfe_es_task.c.

References CFE_ES_GetMemPoolStats(), CFE_ES_INVALID_POOL_HANDLE_ERR_EID, CFE_ES_TLM_POOL_STATS_INFO_EID, CFE_ES_ValidateHandle(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GET_MEMADDR, CFE_SB_SendMsg(), CFE_SB_SET_MEMADDR, CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_TaskData_t::MemStatsPacket, CFE_ES_SendMemPoolStats_t::Payload, CFE_ES_MemStatsTlm_t::Payload, CFE_ES_SendMemPoolStatsCmd_Payload_t::PoolHandle, CFE_ES_PoolStatsTlm_Payload_t::PoolHandle, and CFE_ES_PoolStatsTlm_Payload_t::PoolStats.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.18 CFE_ES_SetMaxPRCountCmd()

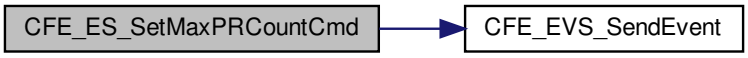
```
int32 CFE_ES_SetMaxPRCountCmd (
            const CFE_ES_SetMaxPRCount_t * data )
```

Definition at line 1725 of file cfe_es_task.c.

References CFE_ES_ResetDataPtr, CFE_ES_SET_MAX_PR_COUNT_EID, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_SetMaxPRCountCmd_Payload_t::MaxPRCount, CFE_ES_ResetVariables_t::MaxProcessorResetCount, CFE_ES_SetMaxPRCount_t::Payload, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.19 CFE_ES_SetPerfFilterMaskCmd()

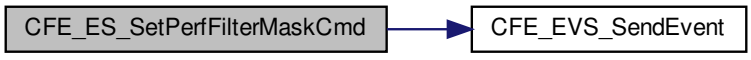
```
int32 CFE_ES_SetPerfFilterMaskCmd (
            const CFE_ES_SetPerfFilterMask_t * data )
```

Definition at line 336 of file cfe_es_perf.c.

References CFE_ES_PERF_32BIT_WORDS_IN_MASK, CFE_ES_PERF_FILTMSKCMD_EID, CFE_ES_PERF_F↩ILTMSKERR_EID, CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EV↩S_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandError↩Counter, CFE_ES_PerfMetaData_t::FilterMask, CFE_ES_SetPerfFilterMaskCmd_Payload_t::FilterMask, CFE_ES↩_SetPerfFilterMaskCmd_Payload_t::FilterMaskNum, CFE_ES_PerfData_t::MetaData, and CFE_ES_SetPerfFilter↩Mask_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.20 CFE_ES_SetPerfTriggerMaskCmd()

```
int32 CFE_ES_SetPerfTriggerMaskCmd (
            const CFE_ES_SetPerfTriggerMask_t * data )
```

Definition at line 366 of file cfe_es_perf.c.

References CFE_ES_PERF_32BIT_WORDS_IN_MASK, CFE_ES_PERF_TRIGMSKCMD_EID, CFE_ES_PERF_TRIGMSKERR_EID, CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_PerfData_t::MetaData, CFE_ES_SetPerfTriggerMask_t::Payload, CFE_ES_PerfMetaData_t::TriggerMask, CFE_ES_SetPerfTrigMaskCmd_Payload_t::TriggerMask, and CFE_ES_SetPerfTrigMaskCmd_Payload_t::TriggerMaskNum.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:

13.34.2.21 CFE_ES_ShellCmd()

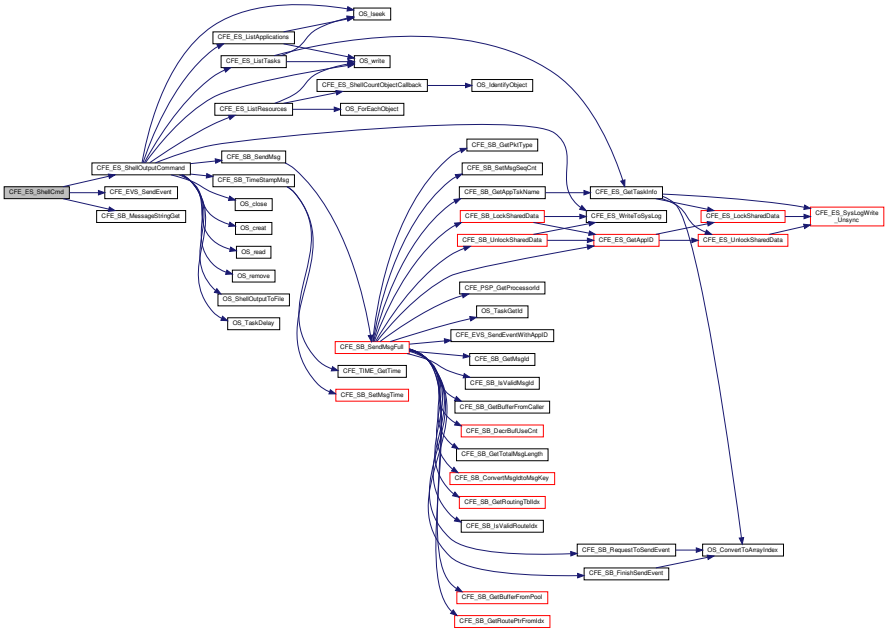
```
int32 CFE_ES_ShellCmd (
            const CFE_ES_Shell_t * data )
```

Definition at line 831 of file cfe_es_task.c.

References CFE_ES_SHELL_ERR_EID, CFE_ES_SHELL_INF_EID, CFE_ES_ShellOutputCommand(), CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_PLATFORM_ES_MAX_SHELL_CMD, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_ShellCmd_Payload_t::CmdString, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, OS_MAX_PATH_LEN, CFE_ES_ShellCmd_Payload_t::OutputFilename, and CFE_ES_Shell_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.22 CFE_ES_StartAppCmd()

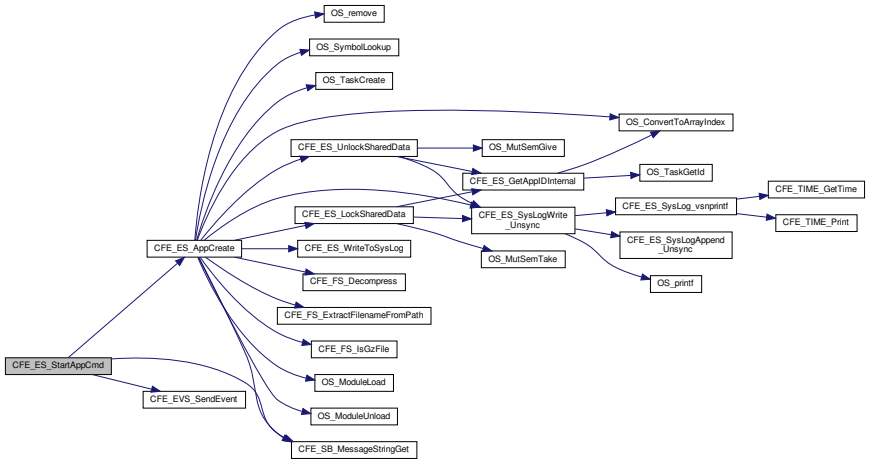
```
int32 CFE_ES_StartAppCmd (
            const CFE_ES_StartApp_t * data )
```

Definition at line 877 of file cfe_es_task.c.

References CFE_ES_StartAppCmd_Payload_t::AppEntryPoint, CFE_ES_StartAppCmd_Payload_t::AppFileName, CFE_ES_StartAppCmd_Payload_t::Application, CFE_ES_AppCreate(), CFE_ES_ExceptionAction_PROC_RESTART, CFE_ES_ExceptionAction_RESTART_APP, CFE_ES_START_ERR_EID, CFE_ES_START_EXC_ACTION_ERR_EID, CFE_ES_START_INF_EID, CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID, CFE_ES_START_INVALID_FILENAME_ERR_EID, CFE_ES_START_NULL_APP_NAME_ERR_EID, CFE_ES_START_PRIORITY_ERR_EID, CFE_ES_START_STACK_ERR_EID, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_PLATFORM_ES_DEFAULT_STACK_SIZE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_StartAppCmd_Payload_t::ExceptionAction, NULL, OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_MAX_PRIORITY, CFE_ES_StartApp_t::Payload, CFE_ES_StartAppCmd_Payload_t::Priority, and CFE_ES_StartAppCmd_Payload_t::StackSize.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.23 CFE_ES_StartPerfDataCmd()

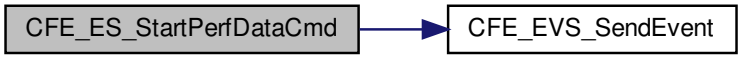
```
int32 CFE_ES_StartPerfDataCmd (
            const CFE_ES_StartPerfData_t * data )
```

Definition at line 126 of file cfe_es_perf.c.

References CFE_ES_PERF_MAX_MODES, CFE_ES_PERF_STARTCMD_EID, CFE_ES_PERF_STARTCMD_E↩ RR_EID, CFE_ES_PERF_STARTCMD_TRIG_ERR_EID, CFE_ES_PERF_TRIGGER_END, CFE_ES_PERF_TR↩ IGGER_START, CFE_ES_PERF_WAITING_FOR_TRIGGER, CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::↩ DataEnd, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfMetaData_t::↩ InvalidMarkerReported, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, CFE_ES_StartPerfData_↩ t::Payload, CFE_ES_PerfMetaData_t::State, CFE_ES_PerfMetaData_t::TriggerCount, and CFE_ES_StartPerfCmd_↩ Payload_t::TriggerMode.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.24 CFE_ES_StopAppCmd()

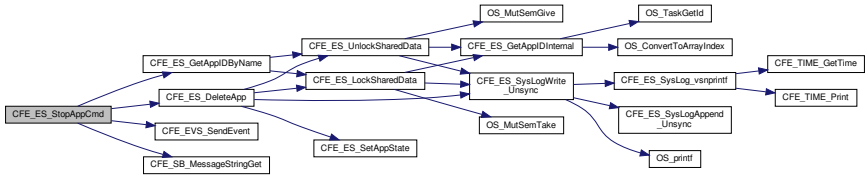
```
int32 CFE_ES_StopAppCmd (
            const CFE_ES_StopApp_t * data )
```

Definition at line 985 of file cfe_es_task.c.

References CFE_ES_AppNameCmd_Payload_t::Application, CFE_ES_DeleteApp(), CFE_ES_GetAppIDByName(), CFE_ES_STOP_DBG_EID, CFE_ES_STOP_ERR1_EID, CFE_ES_STOP_ERR2_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, NULL, OS_MAX_API_NAME, and CFE_ES_AppNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.25 CFE_ES_StopPerfDataCmd()

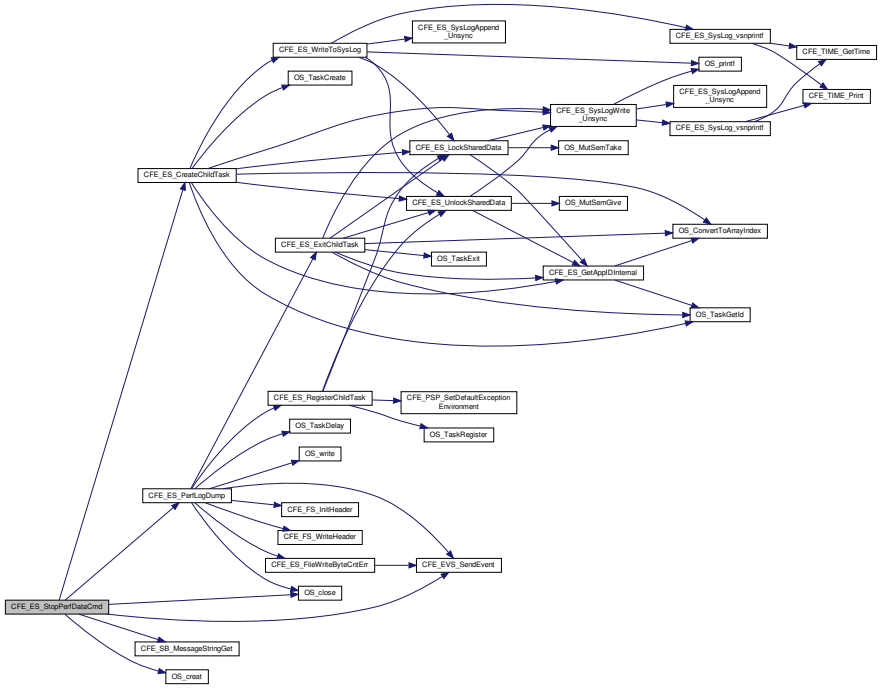
```
int32 CFE_ES_StopPerfDataCmd (
            const CFE_ES_StopPerfData_t * data )
```

Definition at line 176 of file cfe_es_perf.c.

References CFE_ES_CreateChildTask(), CFE_ES_PERF_CHILD_FLAGS, CFE_ES_PERF_CHILD_NAME, CFE_ES_PERF_CHILD_STACK_PTR, CFE_ES_PERF_IDLE, CFE_ES_PERF_LOG_ERR_EID, CFE_ES_PERF_STOPCMD_EID, CFE_ES_PERF_STOPCMD_ERR1_EID, CFE_ES_PERF_STOPCMD_ERR2_EID, CFE_ES_PerfLogDump(), CFE_ES_TaskData, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME, CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY, CFE_PLATFORM_ES_PERF_CHILD_PRIORITY, CFE_PLATFORM_ES_PERF_CHILD_STACK_SIZE, CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_PerfLogDump_t::ChildID, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfLogDump_t::DataFileDescriptor, CFE_ES_PerfLogDump_t::DataFileName, CFE_ES_StopPerfCmd_Payload_t::DataFileName, CFE_ES_PerfLogDump_t::DataToWrite, CFE_ES_PerfData_t::MetaData, OS_close(), OS_creat(), OS_MAX_PATH_LEN, OS_WRITE_ONLY, CFE_ES_StopPerfData_t::Payload, and CFE_ES_PerfMetaData_t::State.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.26 CFE_ES_TaskInit()

```
int32 CFE_ES_TaskInit (
            void  )
```

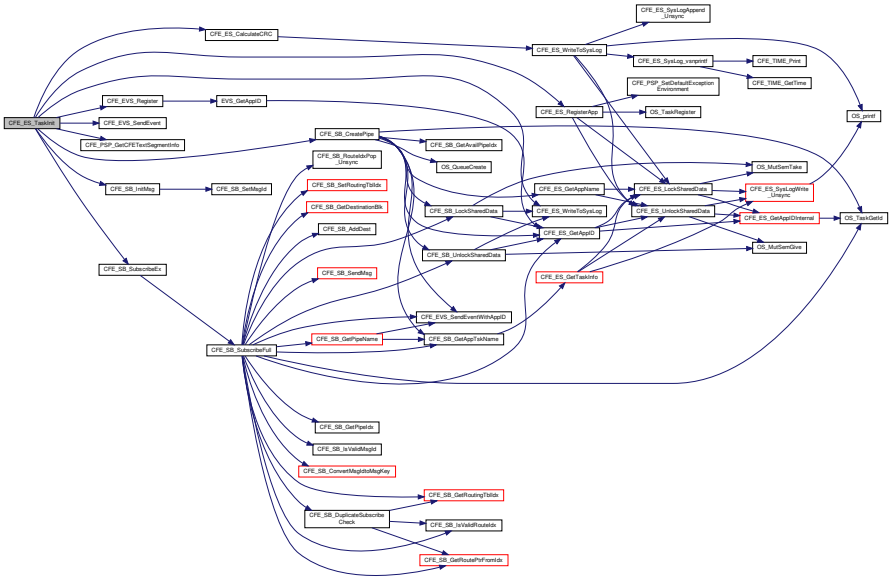
Definition at line 203 of file cfe_es_task.c.

References CFE_ES_APP_TLM_MID, CFE_ES_BUILD_INF_EID, CFE_ES_CalculateCRC(), CFE_ES_CMD_M↩ID, CFE_ES_HK_TLM_MID, CFE_ES_INIT_INF_EID, CFE_ES_INITSTATS_INF_EID, CFE_ES_MEMSTATS_TL↩M_MID, CFE_ES_RegisterApp(), CFE_ES_ResetDataPtr, CFE_ES_SEND_HK_MID, CFE_ES_SHELL_TLM_MID, CFE_ES_VERSION_INF_EID, CFE_ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType↩_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_MAJOR_VERSION, CFE_MINOR_VERS↩ION, CFE_MISSION_ES_DEFAULT_CRC, CFE_MISSION_EVS_MAX_MESSAGE_LENGTH, CFE_MISSION_REV, CFE_PLATFORM_ES_DEFAULT_SYSLOG_MODE, CFE_PSP_GetCFETextSegmentInfo(), CFE_PSP_MAJOR_V↩ERSION, CFE_PSP_MINOR_VERSION, CFE_PSP_MISSION_REV, CFE_PSP_REVISION, CFE_PSP_SUCCESS, CFE_REVISION, CFE_SB_CreatePipe(), CFE_SB_Default_Qos, CFE_SB_InitMsg(), CFE_SB_SubscribeEx(), CFE↩_SUCCESS, CFE_ES_HousekeepingTlm_Payload_t::CFECoreChecksum, CFE_ES_HousekeepingTlm_Payload_t::↩CFEMajorVersion, CFE_ES_HousekeepingTlm_Payload_t::CFEMinorVersion, CFE_ES_HousekeepingTlm_Payload↩_t::CFEMissionRevision, CFE_ES_HousekeepingTlm_Payload_t::CFERevision, CFE_ES_TaskData_t::CmdPipe, C↩FE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_TaskData_t::Hk↩Packet, CFE_ES_TaskData_t::LimitCmd, CFE_ES_TaskData_t::LimitHK, CFE_ES_TaskData_t::MemStatsPacket, N↩ULL, CFE_ES_TaskData_t::OneAppPacket, OS_MAJOR_VERSION, OS_MINOR_VERSION, OS_MISSION_REV,

OS_REVISION, CFE_ES_HousekeepingTlm_Payload_t::OSALMajorVersion, CFE_ES_HousekeepingTlm_Payload_t::OSALMinorVersion, CFE_ES_HousekeepingTlm_Payload_t::OSALMissionRevision, CFE_ES_HousekeepingTlm_Payload_t::OSALRevision, CFE_ES_HousekeepingTlm_t::Payload, CFE_ES_TaskData_t::PipeDepth, CFE_ES_TaskData_t::PipeName, CFE_ES_TaskData_t::ShellPacket, and CFE_ES_ResetData_t::SystemLogMode.

Referenced by CFE_ES_TaskMain().

Here is the call graph for this function:



13.34.2.27 CFE_ES_TaskMain()

```
void CFE_ES_TaskMain (
            void  )
```

Description

This is the entry point to the cFE ES Core Application.

Assumptions, External Events, and Notes:

None

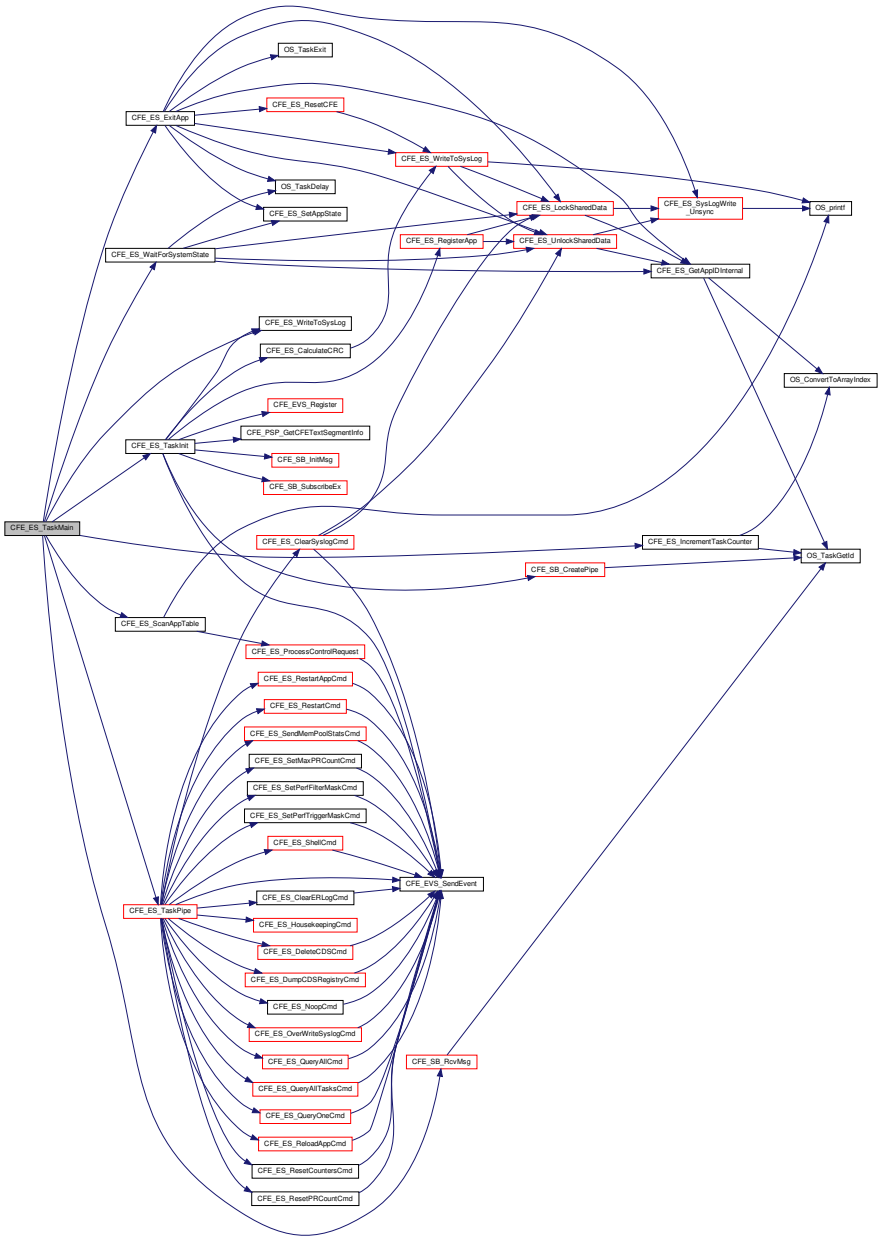
Return values

None	

Definition at line 80 of file cfe_es_task.c.

References CFE_ES_ExitApp(), CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunStatus_APP_RUN, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_A↩ PP_RUNTIME_ERROR, CFE_ES_ScanAppTable(), CFE_ES_SystemState_CORE_READY, CFE_ES_TaskInit(), CF↩ E_ES_TaskPipe(), CFE_ES_WaitForSystemState(), CFE_ES_WriteToSysLog(), CFE_MISSION_ES_MAIN_PERF_ID, CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_PLATFORM_ES_APP_SCAN_RATE, CFE_SB_RcvMsg(), CFE_SB_TIME_OUT, CFE_SUCCESS, CFE_ES_TaskData_t::CmdPipe, and CFE_ES_TaskData_t::MsgPtr.

Here is the call graph for this function:



13.34.2.28 CFE_ES_TaskPipe()

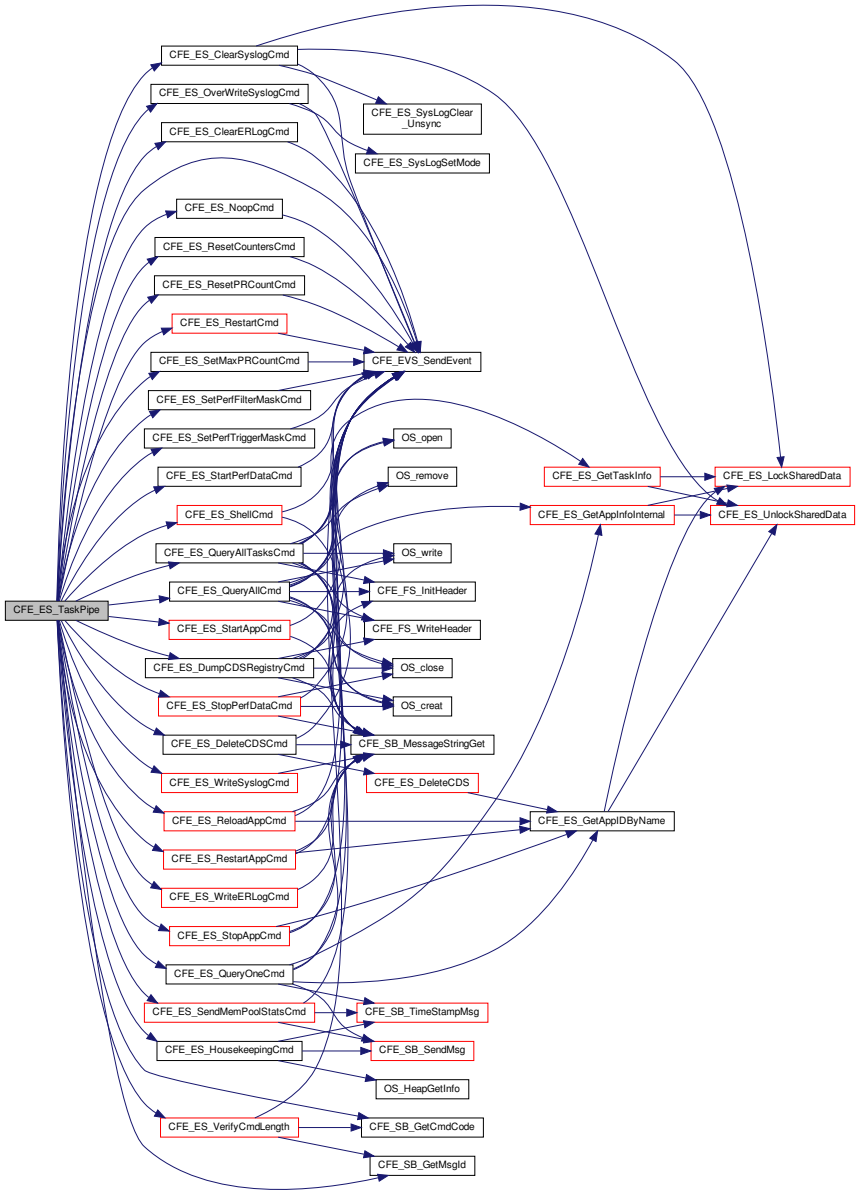
```
void CFE_ES_TaskPipe (
            CFE_SB_MsgPtr_t Msg )
```

Definition at line 413 of file cfe_es_task.c.

References CFE_ES_CC1_ERR_EID, CFE_ES_CLEAR_ER_LOG_CC, CFE_ES_CLEAR_SYSLOG_CC, CFE_ES_ClearERLogCmd(), CFE_ES_ClearSyslogCmd(), CFE_ES_CMD_MID, CFE_ES_DELETE_CDS_CC, CFE_ES_DeleteCDSCmd(), CFE_ES_DUMP_CDS_REGISTRY_CC, CFE_ES_DumpCDSRegistryCmd(), CFE_ES_HousekeepingCmd(), CFE_ES_MID_ERR_EID, CFE_ES_NOOP_CC, CFE_ES_NoopCmd(), CFE_ES_OVER_WRITE_SYSLOG_CC, CFE_ES_OverWriteSyslogCmd(), CFE_ES_QUERY_ALL_CC, CFE_ES_QUERY_ALL_TASKS_CC, CFE_ES_QUERY_ONE_CC, CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_RELOAD_APP_CC, CFE_ES_ReloadAppCmd(), CFE_ES_RESET_COUNTERS_CC, CFE_ES_RESET_PR_COUNT_CC, CFE_ES_ResetCountersCmd(), CFE_ES_ResetPRCountCmd(), CFE_ES_RESTART_APP_CC, CFE_ES_RESTART_CC, CFE_ES_RestartAppCmd(), CFE_ES_RestartCmd(), CFE_ES_SEND_HK_MID, CFE_ES_SEND_MEM_POOL_STATS_CC, CFE_ES_SendMemPoolStatsCmd(), CFE_ES_SET_MAX_PR_COUNT_CC, CFE_ES_SET_PERF_FILTER_MASK_CC, CFE_ES_SET_PERF_TRIGGER_MASK_CC, CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_SHELL_CC, CFE_ES_ShellCmd(), CFE_ES_START_APP_CC, CFE_ES_START_PERF_DATA_CC, CFE_ES_StartAppCmd(), CFE_ES_StartPerfDataCmd(), CFE_ES_STOP_APP_CC, CFE_ES_STOP_PERF_DATA_CC, CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_VerifyCmdLength(), CFE_ES_WRITE_ER_LOG_CC, CFE_ES_WRITE_SYSLOG_CC, CFE_ES_WriteERLogCmd(), CFE_ES_WriteSyslogCmd(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(), and CFE_ES_TaskData_t::CommandErrorCounter.

Referenced by CFE_ES_TaskMain().

Here is the call graph for this function:



13.34.2.29 CFE_ES_ValidateHandle()

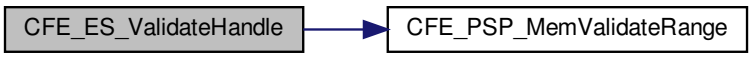
```
bool CFE_ES_ValidateHandle (
            CFE_ES_MemHandle_t Handle )
```

Definition at line 666 of file cfe_esmempool.c.

References CFE_PSP_MEM_ANY, CFE_PSP_MemValidateRange(), CFE_PSP_SUCCESS, Pool_t::End, NULL, Pool_t::PoolHandle, and Pool_t::Size.

Referenced by CFE_ES_SendMemPoolStatsCmd().

Here is the call graph for this function:



13.34.2.30 CFE_ES_VerifyCmdLength()

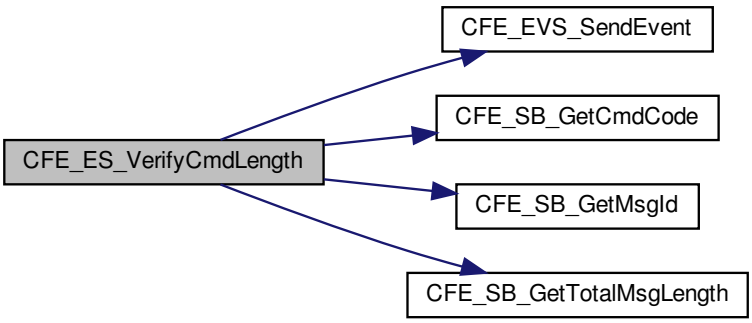
```
bool CFE_ES_VerifyCmdLength (
            CFE_SB_MsgPtr_t msg,
            uint16 ExpectedLength )
```

Definition at line 1669 of file cfe_es_task.c.

References CFE_ES_LEN_ERR_EID, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GetCmd↩Code(), CFE_SB_GetMsgId(), CFE_SB_GetTotalMsgLength(), and CFE_ES_TaskData_t::CommandErrorCounter.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.31 CFE_ES_WriteERLogCmd()

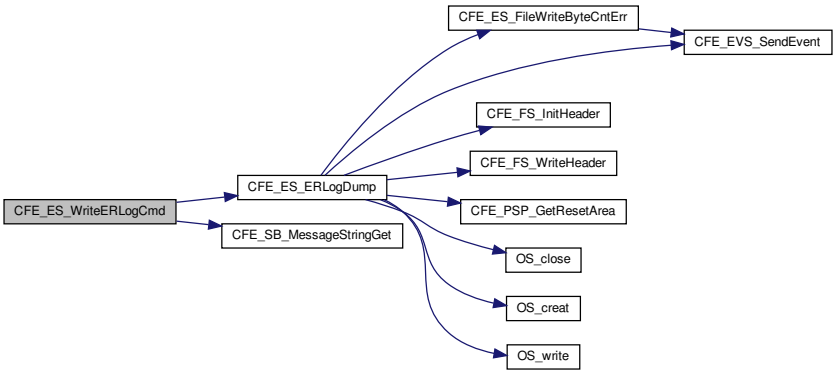
```
int32 CFE_ES_WriteERLogCmd (
            const CFE_ES_WriteERLog_t * data )
```

Definition at line 1571 of file cfe_es_task.c.

References CFE_ES_ERLogDump(), CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_MAX_PATH_LEN, and CFE_ES_FileNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.2.32 CFE_ES_WriteSyslogCmd()

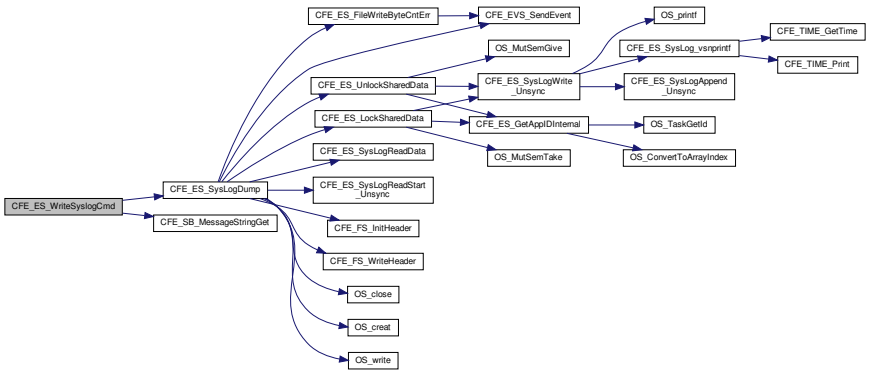
```
int32 CFE_ES_WriteSyslogCmd (
            const CFE_ES_WriteSyslog_t * data )
```

Definition at line 1506 of file cfe_es_task.c.

References CFE_ES_SysLogDump(), CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_ES_TaskData_t::CommandCounter, CFE_ES_TaskData_t::CommandErrorCounter, CFE_ES_FileNameCmd_Payload_t::FileName, OS_MAX_PATH_LEN, and CFE_ES_FileNameCmd_t::Payload.

Referenced by CFE_ES_TaskPipe().

Here is the call graph for this function:



13.34.3 Variable Documentation

13.34.3.1 CFE_ES_TaskData

```
CFE_ES_TaskData_t CFE_ES_TaskData
```

Definition at line 72 of file cfe_es_task.c.

Referenced by CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_ShellOutput↩Command(), CFE_ES_StartPerfDataCmd(), CFE_ES_StopPerfDataCmd(), and CFE_ES_SysLogDump().

13.35 cfe/fsw/cfe-core/src/es/cfe_es_verify.h File Reference

```
#include <stdint.h>
```

13.36 cfe/fsw/cfe-core/src/es/cfe_esmempool.c File Reference

```
#include "private/cfe_private.h"
#include "cfe_esmempool.h"
#include "cfe_es.h"
#include "cfe_es_task.h"
#include "cfe_es_log.h"
#include <stdio.h>
```

Data Structures

- union MemPoolAddr_t

Macros

- #define ALIGN_OF(type) ((cpuaddr)&((struct { char Byte; type Align; } *)0)->Align)
- #define CFE_ES_CHECK_PATTERN 0x5a5a
- #define CFE_ES_MEMORY_ALLOCATED 0xaaaa
- #define CFE_ES_MEMORY_DEALLOCATED 0xdddd

Functions

- uint32 CFE_ES_GetBlockSize (Pool_t *PoolPtr, uint32 Size)
- int32 CFE_ES_PoolCreateNoSem (CFE_ES_MemHandle_t *HandlePtr, uint8 *MemPtr, uint32 Size)

  Initializes a memory pool created by an application without using a semaphore during processing.
- int32 CFE_ES_PoolCreate (CFE_ES_MemHandle_t *HandlePtr, uint8 *MemPtr, uint32 Size)

  Initializes a memory pool created by an application while using a semaphore during processing.
- int32 CFE_ES_PoolCreateEx (CFE_ES_MemHandle_t *HandlePtr, uint8 *MemPtr, uint32 Size, uint32 Num↩ BlockSizes, uint32 *BlockSizes, uint16 UseMutex)

  Initializes a memory pool created by an application with application specified block sizes.
- int32 CFE_ES_GetPoolBuf (uint32 **BufPtr, CFE_ES_MemHandle_t Handle, uint32 Size)

  Gets a buffer from the memory pool created by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
- int32 CFE_ES_GetPoolBufInfo (CFE_ES_MemHandle_t Handle, uint32 *BufPtr)

  Gets info on a buffer previously allocated via CFE_ES_GetPoolBuf.
- int32 CFE_ES_PutPoolBuf (CFE_ES_MemHandle_t Handle, uint32 *BufPtr)

  Releases a buffer from the memory pool that was previously allocated via CFE_ES_GetPoolBuf.
- int32 CFE_ES_GetMemPoolStats (CFE_ES_MemPoolStats_t *BufPtr, CFE_ES_MemHandle_t Handle)

  Extracts the statistics maintained by the memory pool software.
- bool CFE_ES_ValidateHandle (CFE_ES_MemHandle_t Handle)

Variables

- uint32 CFE_ES_MemPoolDefSize [CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]

13.36.1 Macro Definition Documentation

13.36.1.1 ALIGN_OF

```
#define ALIGN_OF(
             type ) ((cpuaddr)&((struct { char Byte; type Align; } *)0)->Align)
```

Macro that determines the native alignment requirement of a specific type

By getting the offset of the structure after following a single char, this effectively gets how much padding the compiler added, which in turn reveals its minimum alignment requirement. (C99 is lacking a standardized "alignof" operator, and this is intended to substitute).

Definition at line 52 of file cfe_esmempool.c.

Referenced by CFE_ES_PoolCreateEx().

13.36.1.2 CFE_ES_CHECK_PATTERN

```
#define CFE_ES_CHECK_PATTERN 0x5a5a
```

Definition at line 74 of file cfe_esmempool.c.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

13.36.1.3 CFE_ES_MEMORY_ALLOCATED

```
#define CFE_ES_MEMORY_ALLOCATED 0xaaaa
```

Definition at line 75 of file cfe_esmempool.c.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), and CFE_ES_PutPoolBuf().

13.36.1.4 CFE_ES_MEMORY_DEALLOCATED

```
#define CFE_ES_MEMORY_DEALLOCATED 0xdddd
```

Definition at line 76 of file cfe_esmempool.c.

Referenced by CFE_ES_PutPoolBuf().

13.36.2 Function Documentation

13.36.2.1 CFE_ES_GetBlockSize()

```
uint32 CFE_ES_GetBlockSize (
            Pool_t * PoolPtr,
            uint32 Size )
```

Definition at line 595 of file cfe_esmempool.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, BlockSizeDesc_t::MaxSize, NULL, Pool_t::SizeDesc, and Pool_t::SizeDescPtr.

Referenced by CFE_ES_GetPoolBuf(), and CFE_ES_PutPoolBuf().

13.36.2.2 CFE_ES_GetMemPoolStats()

```
int32 CFE_ES_GetMemPoolStats (
            CFE_ES_MemPoolStats_t * BufPtr,
            CFE_ES_MemHandle_t Handle )
```

Description

This routine fills the CFE_ES_MemPoolStats_t data structure with the statistics maintained by the memory pool software. These statistics can then be telemetered by the calling Application.

Assumptions, External Events, and Notes:

None

Parameters

in	*BufPtr*	Pointer to CFE_ES_MemPoolStats_t data structure to be filled with memory statistics.
in	*Handle*	The handle to the memory pool whose statistics are desired.
out	∗*BufPtr*	Memory Pool Statistics stored in given data structure.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.

Returns

See also

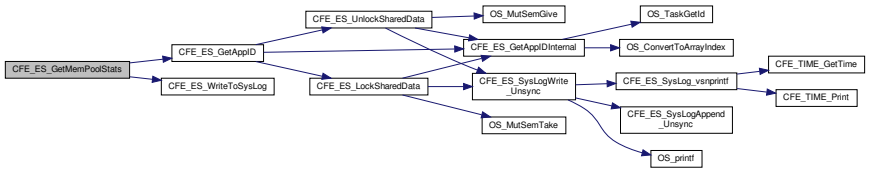
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_↩ PutPoolBuf

Definition at line 627 of file cfe_esmempool.c.

References CFE_ES_BlockStats_t::BlockSize, CFE_ES_MemPoolStats_t::BlockStats, CFE_ES_ERR_MEM_HAND↩LE, CFE_ES_GetAppID(), CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_WriteToSysLog(), CFE_SUCCE↩SS, Pool_t::CheckErrCntr, CFE_ES_MemPoolStats_t::CheckErrCtr, Pool_t::CurrentAddr, Pool_t::End, BlockSizeDesc↩_t::MaxSize, NULL, CFE_ES_MemPoolStats_t::NumBlocksRequested, BlockSizeDesc_t::NumCreated, CFE_ES_↩BlockStats_t::NumCreated, BlockSizeDesc_t::NumFree, CFE_ES_BlockStats_t::NumFree, CFE_ES_MemPoolStats↩_t::NumFreeBytes, Pool_t::PoolHandle, CFE_ES_MemPoolStats_t::PoolSize, Pool_t::RequestCntr, Pool_t::Size, and Pool_t::SizeDesc.

Referenced by CFE_ES_SendMemPoolStatsCmd().

Here is the call graph for this function:



13.36.2.3 CFE_ES_GetPoolBuf()

```
int32 CFE_ES_GetPoolBuf (
            uint32 ** BufPtr,
            CFE_ES_MemHandle_t HandlePtr,
            uint32 Size )
```

Description

This routine obtains a block of memory from the memory pool supplied by the calling application.

Assumptions, External Events, and Notes:

1. The size allocated from the memory pool is, at a minimum, 12 bytes more than requested.

Parameters

in	*BufPtr*	A pointer to the Application's pointer in which will be stored the address of the allocated memory buffer.
in	*HandlePtr*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*Size*	The size of the buffer requested. NOTE: The size allocated may be larger.
out	∗*BufPtr*	The address of the requested buffer.

When successful, the return value is a positive number and is the number of bytes actually allocated for the buffer.	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.
CFE_ES_ERR_MEM_BLOCK_SIZE	The block size requested is invalid.

Returns

See also

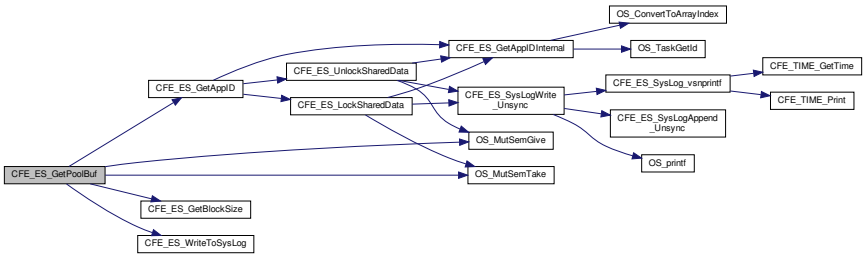
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_PutPoolBuf, CFE_ES_GetMemPoolStats, CFE_ES_GetPoolBufInfo

Definition at line 307 of file cfe_esmempool.c.

References MemPoolAddr_t::Addr, Pool_t::AlignMask, BD::Allocated, MemPoolAddr_t::BdPtr, CFE_ES_CHECK_PATTERN, CFE_ES_ERR_MEM_BLOCK_SIZE, CFE_ES_ERR_MEM_HANDLE, CFE_ES_GetAppID(), CFE_ES_GetBlockSize(), CFE_ES_MEMORY_ALLOCATED, CFE_ES_USE_MUTEX, CFE_ES_WriteToSysLog(), BD::CheckBits, Pool_t::CurrentAddr, Pool_t::End, BlockSizeDesc_t::MaxSize, Pool_t::MutexId, BD::Next, NULL, BlockSizeDesc_t::NumCreated, BlockSizeDesc_t::NumFree, OS_MutSemGive(), OS_MutSemTake(), Pool_t::PoolHandle, Pool_t::RequestCntr, BD::Size, Pool_t::Size, Pool_t::SizeDesc, Pool_t::SizeDescPtr, BlockSizeDesc_t::Top, Pool_t::UseMutex, and MemPoolAddr_t::UserPtr.

Referenced by CFE_SB_AppInit(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), and CFE_SB_ZeroCopyGetPtr().

Here is the call graph for this function:



13.36.2.4 CFE_ES_GetPoolBufInfo()

```
int32 CFE_ES_GetPoolBufInfo (
            CFE_ES_MemHandle_t HandlePtr,
            uint32 * BufPtr )
```

Description

This routine gets info on a buffer in the memory pool.

Assumptions, External Events, and Notes:

None

Parameters

in	*HandlePtr*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*BufPtr*	A pointer to the memory buffer to provide status for.

When successful, the return value is a positive number and is the number of bytes actually allocated.	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.
CFE_ES_BUFFER_NOT_IN_POOL	The specified address is not in the memory pool.

Returns

See also

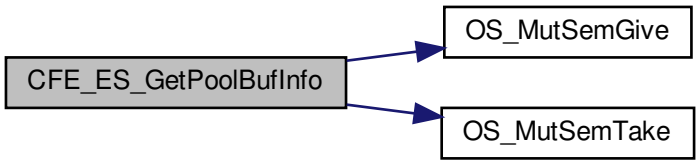
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_GetMemPoolStats, CFE_ES_PutPoolBuf

Definition at line 435 of file cfe_esmempool.c.

References MemPoolAddr_t::Addr, BD::Allocated, MemPoolAddr_t::BdPtr, CFE_ES_BUFFER_NOT_IN_POOL, CFE_ES_CHECK_PATTERN, CFE_ES_ERR_MEM_HANDLE, CFE_ES_MEMORY_ALLOCATED, CFE_ES_USE_MUTEX, BD::CheckBits, Pool_t::End, Pool_t::MutexId, NULL, OS_MutSemGive(), OS_MutSemTake(), Pool_t::PoolHandle, BD::Size, Pool_t::UseMutex, and MemPoolAddr_t::UserPtr.

Referenced by CFE_SB_ZeroCopyReleaseDesc().

Here is the call graph for this function:



13.36.2.5 CFE_ES_PoolCreate()

```
int32 CFE_ES_PoolCreate (
            CFE_ES_MemHandle_t * HandlePtr,
            uint8 * MemPtr,
            uint32 Size )
```

Description

This routine initializes a pool of memory supplied by the calling application. When a memory pool created by this routine is processed, mutex handling will be performed.

Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

Parameters

in	*HandlePtr*	A pointer to the variable the caller wishes to have the memory pool handle kept in.
in	*MemPtr*	A Pointer to the pool of memory created by the calling application. This address must be on a 32-bit boundary.
in	*Size*	The size of the pool of memory. Note that this must be an integral number of 32 bit words.
out	∗*HandlePtr*	The memory pool handle.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

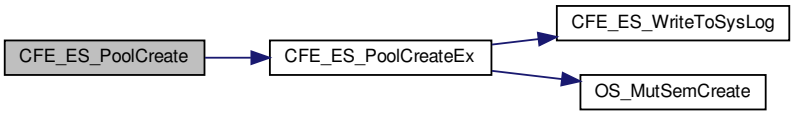
See also

CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf, CFE_ES_↩ GetMemPoolStats

Definition at line 129 of file cfe_esmempool.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_MemPoolDefSize, CFE_ES_PoolCreateEx(), and CFE_ES_USE_MUTEX.

Here is the call graph for this function:



13.36.2.6 CFE_ES_PoolCreateEx()

```
int32 CFE_ES_PoolCreateEx (
            CFE_ES_MemHandle_t * HandlePtr,
            uint8 * MemPtr,
            uint32 Size,
            uint32 NumBlockSizes,
            uint32 * BlockSizes,
            uint16 UseMutex )
```

Description

This routine initializes a pool of memory supplied by the calling application.

Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

Parameters

in	*HandlePtr*	A pointer to the variable the caller wishes to have the memory pool handle kept in.
in	*MemPtr*	A Pointer to the pool of memory created by the calling application. This address must be on a 32-bit boundary.
in	*Size*	The size of the pool of memory. Note that this must be an integral number of 32 bit words.
in	*NumBlockSizes*	The number of different block sizes specified in the `BlockSizes` array. If set equal to zero or if greater than 17, then default block sizes are used.
in	*BlockSizes*	Pointer to an array of sizes to be used instead of the default block sizes specified by CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 through CFE_PLATFORM_ES_MAX_BLOCK_SIZE. If the pointer is equal to NULL, the default block sizes are used.
in	*UseMutex*	Flag indicating whether the new memory pool will be processing with mutex handling or not. Valid parameter values are CFE_ES_USE_MUTEX and CFE_ES_NO_MUTEX
out	∗*HandlePtr*	The memory pool handle.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

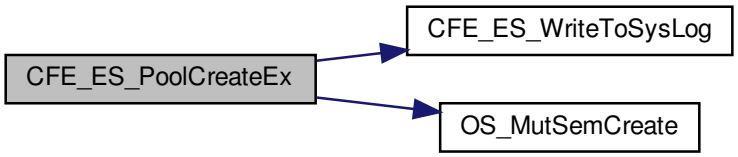
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf, CFE_ES_GetMemPoolStats

Definition at line 138 of file cfe_esmempool.c.

References ALIGN_OF, Pool_t::AlignMask, CFE_ES_BAD_ARGUMENT, CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_MemPoolDefSize, CFE_ES_NO_MUTEX, CFE_ES_STATIC_POOL_TYPE, CFE_ES_USE_MUTEX, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN, CFE_SUCCESS, Pool_t::CheckErrCntr, Pool_t::CurrentAddr, Pool_t::End, BlockSizeDesc_t::MaxSize, Pool_t::MutexId, NULL, BlockSizeDesc_t::NumCreated, BlockSizeDesc_t::NumFree, OS_MAX_API_NAME, OS_MutSemCreate(), Pool_t::PoolHandle, Pool_t::RequestCntr, Pool_t::Size, Pool_t::SizeDesc, Pool_t::SizeDescPtr, BlockSizeDesc_t::Top, and Pool_t::UseMutex.

Referenced by CFE_ES_PoolCreate(), CFE_ES_PoolCreateNoSem(), and CFE_SB_InitBuffers().

Here is the call graph for this function:



13.36.2.7 CFE_ES_PoolCreateNoSem()

```
int32 CFE_ES_PoolCreateNoSem (
            CFE_ES_MemHandle_t * HandlePtr,
            uint8 * MemPtr,
            uint32 Size )
```

Description

This routine initializes a pool of memory supplied by the calling application. When a memory pool created by this routine is processed, no mutex handling is performed.

Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

Parameters

in	*HandlePtr*	A pointer to the variable the caller wishes to have the memory pool handle kept in.
in	*MemPtr*	A Pointer to the pool of memory created by the calling application. This address must be on a 32-bit boundary.
in	*Size*	The size of the pool of memory. Note that this must be an integral number of 32 bit words.
out	∗*HandlePtr*	The memory pool handle.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_PoolCreate, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf, CFE_ES_Get↩MemPoolStats

Definition at line 118 of file cfe_esmempool.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_MemPoolDefSize, CFE_ES_NO_MUTEX, and C↩FE_ES_PoolCreateEx().

Here is the call graph for this function:

13.36.2.8 CFE_ES_PutPoolBuf()

```
int32 CFE_ES_PutPoolBuf (
            CFE_ES_MemHandle_t HandlePtr,
            uint32 * BufPtr )
```

Description

This routine releases a buffer back into the memory pool.

Assumptions, External Events, and Notes:

None

Parameters

in	*HandlePtr*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*BufPtr*	A pointer to the memory buffer to be released.

When successful, the return value is a positive number and is the number of bytes actually released.	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.

Returns

See also

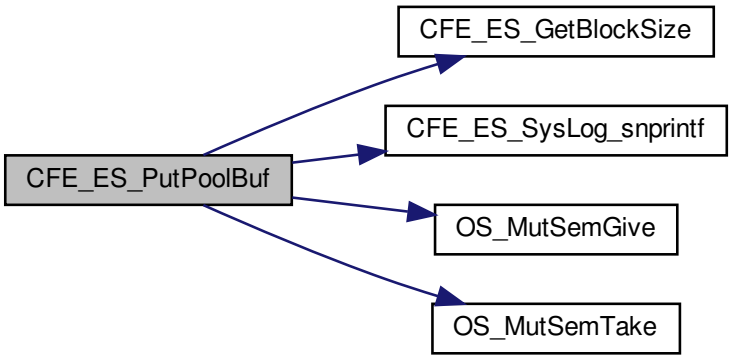
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_GetMemPoolStats, CFE_ES_GetPoolBufInfo

Definition at line 492 of file cfe_esmempool.c.

References MemPoolAddr_t::Addr, BD::Allocated, MemPoolAddr_t::BdPtr, CFE_ES_CHECK_PATTERN, CFE_ES_ERR_MEM_HANDLE, CFE_ES_GetBlockSize(), CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_MEMORY_ALLOCATED, CFE_ES_MEMORY_DEALLOCATED, CFE_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_ES_USE_MUTEX, BD::CheckBits, Pool_t::CheckErrCntr, Pool_t::End, BlockSizeDesc_t::MaxSize, Pool_t::MutexId, BD::Next, NULL, BlockSizeDesc_t::NumFree, OS_MutSemGive(), OS_MutSemTake(), Pool_t::PoolHandle, BD::Size, Pool_t::SizeDesc, Pool_t::SizeDescPtr, BlockSizeDesc_t::Top, Pool_t::UseMutex, and MemPoolAddr_t::UserPtr.

Referenced by CFE_SB_AppInit(), CFE_SB_PutDestinationBlk(), CFE_SB_ReturnBufferToPool(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

Here is the call graph for this function:



13.36.2.9 CFE_ES_ValidateHandle()

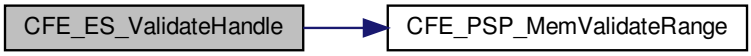
```
bool CFE_ES_ValidateHandle (
            CFE_ES_MemHandle_t Handle )
```

Definition at line 666 of file cfe_esmempool.c.

References CFE_PSP_MEM_ANY, CFE_PSP_MemValidateRange(), CFE_PSP_SUCCESS, Pool_t::End, NULL, Pool_t::PoolHandle, and Pool_t::Size.

Referenced by CFE_ES_SendMemPoolStatsCmd().

Here is the call graph for this function:



13.36.3 Variable Documentation

13.36.3.1 CFE_ES_MemPoolDefSize

```
uint32 CFE_ES_MemPoolDefSize[CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]
```

Initial value:

```
=
{
    CFE_PLATFORM_ES_MAX_BLOCK_SIZE,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_16,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_15,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_14,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_13,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_12,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_11,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_10,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_09,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_08,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02,
    CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01
}
```

Definition at line 83 of file cfe_esmempool.c.

Referenced by CFE_ES_PoolCreate(), CFE_ES_PoolCreateEx(), and CFE_ES_PoolCreateNoSem().

13.37 cfe/fsw/cfe-core/src/es/cfe_esmempool.h File Reference

```
#include "common_types.h"
```

Data Structures

- struct BD
- struct BlockSizeDesc_t
- struct Pool_t

Typedefs

- typedef struct BD BD_t

13.37.1 Detailed Description

Created on: Jan 21, 2015 Author: joseph.p.hickey@nasa.gov

Contains data structure definitions used by the ES mempool implementation. These had previously been defined in cfe_esmempool.c. The definitions are moved into this header file so they can be shared with the unit test.

13.37.2 Typedef Documentation

13.37.2.1 BD_t

```
typedef struct BD BD_t
```

Definition at line 40 of file cfe_esmempool.h.

13.38 cfe/fsw/cfe-core/src/evs/cfe_evs.c File Reference

```
#include "cfe_evs.h"
#include "cfe_evs_task.h"
#include "cfe_evs_utils.h"
#include "common_types.h"
#include "cfe_es.h"
#include "cfe_error.h"
#include <stdarg.h>
#include <string.h>
```

Functions

- int32 CFE_EVS_Register (void ∗Filters, uint16 NumEventFilters, uint16 FilterScheme)

  Register an application for receiving event services.
- int32 CFE_EVS_Unregister (void)

  Cleanup internal structures used by the event manager for the calling Application.
- int32 CFE_EVS_SendEvent (uint16 EventID, uint16 EventType, const char ∗Spec,...)
- int32 CFE_EVS_SendEventWithAppID (uint16 EventID, uint16 EventType, uint32 AppID, const char ∗Spec,...)
- int32 CFE_EVS_SendTimedEvent (CFE_TIME_SysTime_t Time, uint16 EventID, uint16 EventType, const char ∗Spec,...)
- int32 CFE_EVS_ResetFilter (int16 EventID)

  Resets the calling application's event filter for a single event ID.
- int32 CFE_EVS_ResetAllFilters (void)

  Resets all of the calling application's event filters.

13.38.1 Function Documentation

13.38.1.1 CFE_EVS_Register()

```
int32 CFE_EVS_Register (
            void * Filters,
            uint16 NumFilteredEvents,
            uint16 FilterScheme )
```

Description

This routine registers an application with event services and allocates/initializes the internal data structures used to support this application's events. An application may not send events unless it has called this routine. The routine also accepts a filter array structure for applications requiring event filtering. In the current implementation of the EVS, only the binary filtering scheme is supported. See section TBD of the cFE Application Programmer's Guide for a description of the behavior of binary filters. Applications may call CFE_EVS_Register more than once, but each call will wipe out all filters registered by previous calls (filter registration is NOT cumulative).

Assumptions, External Events, and Notes:

Note: Event filters can be added, deleted or modified by ground commands. All filtering schemes include a default setting that results in no filtering (such as CFE_EVS_NO_FILTER for binary filters).

Filter Scheme: Binary
Code: CFE_EVS_EventFilter_BINARY
Filter Structure:

```
typedef struct {
      uint16  EventID,
      uint16  Mask ;
} CFE_EVS_BinFilter_t;
```

Parameters

in	*Filters*	Pointer to an array of event message filters, or NULL if no filtering is desired. The structure of an event message filter depends on the FilterScheme selected. (see Filter Schemes mentioned above)
in	*NumFilteredEvents*	The number of event message filters included in this call. This must be less than or equal to the maximum number of events allowed per application (CFE_PLATFORM_EVS_MAX_EVENT_FILTERS).
in	*FilterScheme*	The event filtering scheme that this application will use. For the first implementation of the event services, only filter type CFE_EVS_EventFilter_BINARY will be supported.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_FILTER_OVERLOAD	Number of Application event filters input upon registration is greater than CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
CFE_EVS_UNKNOWN_FILTER	CFE_EVS_Register() FilterScheme parameter was illegal
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	

Returns

See also

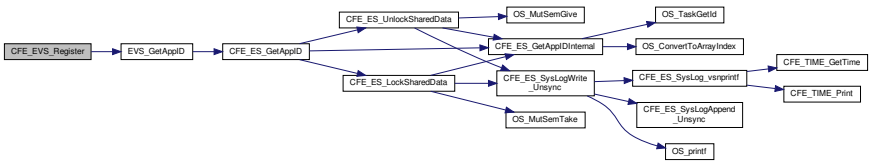
CFE_EVS_Unregister

Definition at line 60 of file cfe_evs.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_ES_ERR_BUFFER, CFE_EVS_EventFilter_BINARY, CFE_EVS_FREE_SLOT, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_EVS_UNKNOWN_FILTER, CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SUCCESS, EVS_BinFilter_t::Count, EVS_AppData_t::EventCount, EVS_BinFilter_t::EventID, CFE_EVS_BinFilter_t::EventID, EVS_AppData_t::EventTypesActiveFlag, EVS_GetAppID(), EVS_BinFilter_t::Mask, CFE_EVS_BinFilter_t::Mask, NULL, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), and CFE_SB_AppInit().

Here is the call graph for this function:



13.38.1.2 CFE_EVS_ResetAllFilters()

```
int32 CFE_EVS_ResetAllFilters (
            void  )
```

Description

This routine resets all the calling application's event filter counters to zero, providing a quick and convenient method for resetting event filters.

Assumptions, External Events, and Notes:

None

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	

Returns

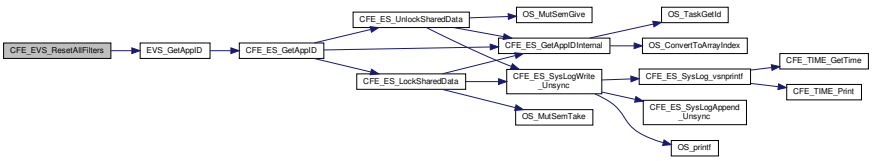
See also

CFE_EVS_ResetFilter

Definition at line 349 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SUCC↩ESS, EVS_BinFilter_t::Count, EVS_GetAppID(), and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



13.38.1.3 CFE_EVS_ResetFilter()

```
int32 CFE_EVS_ResetFilter (
            int16 EventID )
```

Description

The effect of resetting an event filter depends on the filter scheme. The CFE_EVS_EventFilter_BINARY scheme resets the filter counter for the specified Event ID.

Assumptions, External Events, and Notes:

None

Parameters

in	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	

Returns

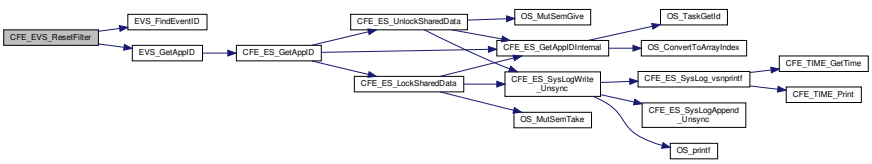
See also

CFE_EVS_ResetAllFilters

Definition at line 304 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_EVT_NOT_REGISTERED, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_SUCCESS, EVS_BinFilter_t::Count, EVS_FindEventID(), EVS_GetAppID(), NULL, and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



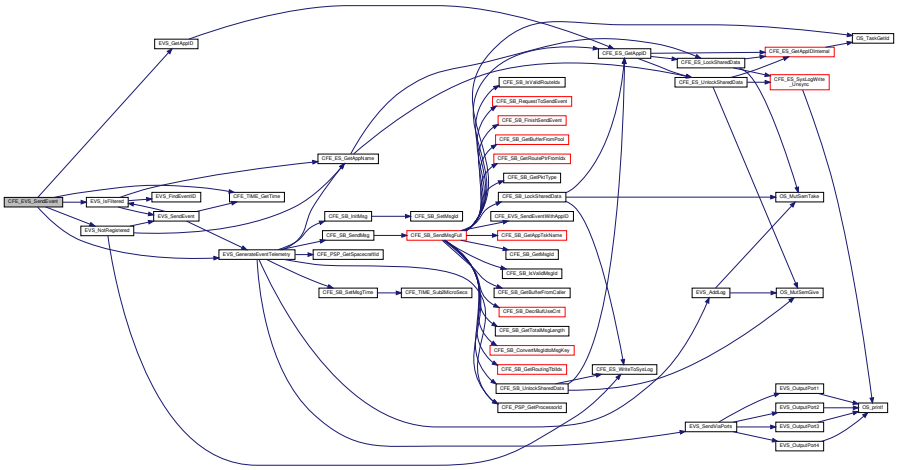
13.38.1.4 CFE_EVS_SendEvent()

```
int32 CFE_EVS_SendEvent (
            uint16 EventID,
            uint16 EventType,
            const char * Spec,
             ... )
```

Definition at line 178 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_SUCCESS, CFE_TIME_GetTime(), EVS_GenerateEventTelemetry(), EVS_GetAppID(), EVS_IsFiltered(), EVS_NotRegistered(), and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



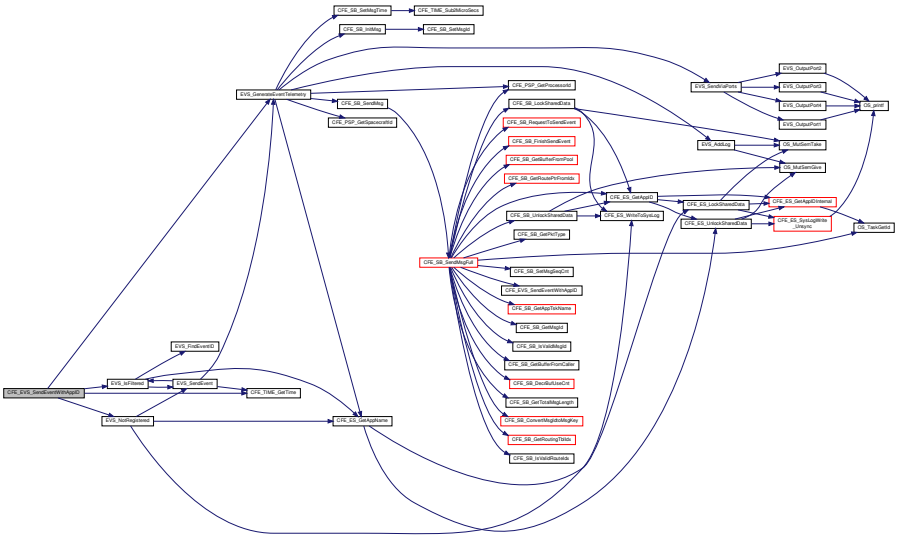
13.38.1.5 CFE_EVS_SendEventWithAppID()

```
int32 CFE_EVS_SendEventWithAppID (
            uint16 EventID,
            uint16 EventType,
            uint32 AppID,
            const char * Spec,
             ... )
```

Definition at line 223 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_GlobalData, CFE↩_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_TIME_GetTime(), EVS_GenerateEventTelemetry(), EVS_IsFiltered(), EVS_NotRegistered(), and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



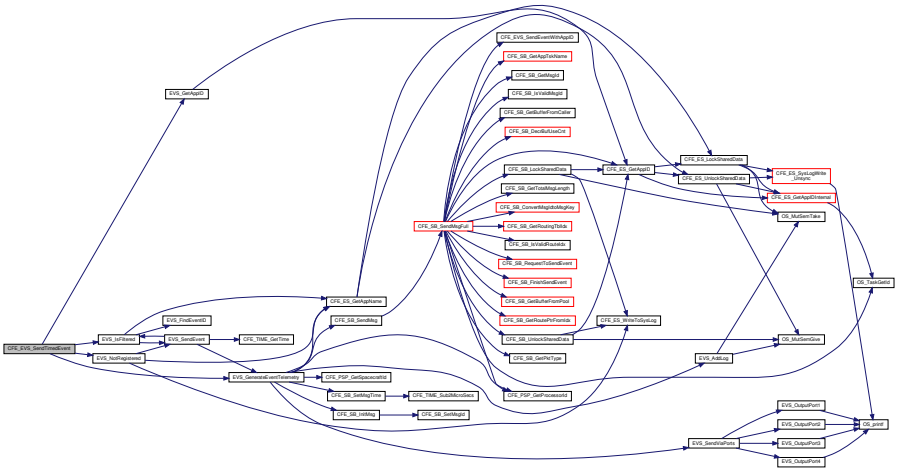
13.38.1.6 CFE_EVS_SendTimedEvent()

```
int32 CFE_EVS_SendTimedEvent (
            CFE_TIME_SysTime_t Time,
            uint16 EventID,
            uint16 EventType,
            const char * Spec,
             ... )
```

Definition at line 264 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_SUCCESS, EVS_GenerateEventTelemetry(), EVS_GetAppID(), EVS_IsFiltered(), EVS_NotRegistered(), and EVS_AppData_t::↩ RegisterFlag.

Here is the call graph for this function:



13.38.1.7 CFE_EVS_Unregister()

```
int32 CFE_EVS_Unregister (
            void  )
```

Description

This routine un-registers the calling application from receiving event services and removes and deletes the calling applications filters and counters from the internal event service filter and counter tables if registered. Applications must call this routine as part of their orderly shutdown process.

Assumptions, External Events, and Notes:

None

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	
Any of the error codes from CFE_ES_PutPoolBuf	

Returns

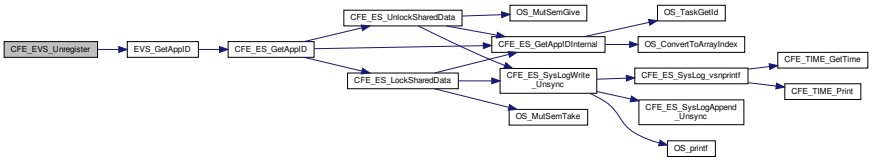
See also

CFE_EVS_Register

Definition at line 146 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_SUCCESS, EVS_GetAppID(), and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



13.39 cfe/fsw/cfe-core/src/evs/cfe_evs.mak File Reference

13.40 cfe/fsw/cfe-core/src/evs/cfe_evs_log.c File Reference

```
#include "cfe_evs_task.h"
#include "cfe_evs_log.h"
#include "cfe_evs.h"
#include "cfe_evs_utils.h"
#include "cfe_fs.h"
#include "cfe_error.h"
#include "cfe_psp.h"
#include <string.h>
```

Functions

- void EVS_AddLog (CFE_EVS_LongEventTlm_t *EVS_PktPtr)
- void EVS_ClearLog (void)
- int32 CFE_EVS_WriteLogDataFileCmd (const CFE_EVS_WriteLogDataFile_t *data)
- int32 CFE_EVS_SetLogModeCmd (const CFE_EVS_SetLogMode_t *data)

13.40.1 Function Documentation

13.40.1.1 CFE_EVS_SetLogModeCmd()

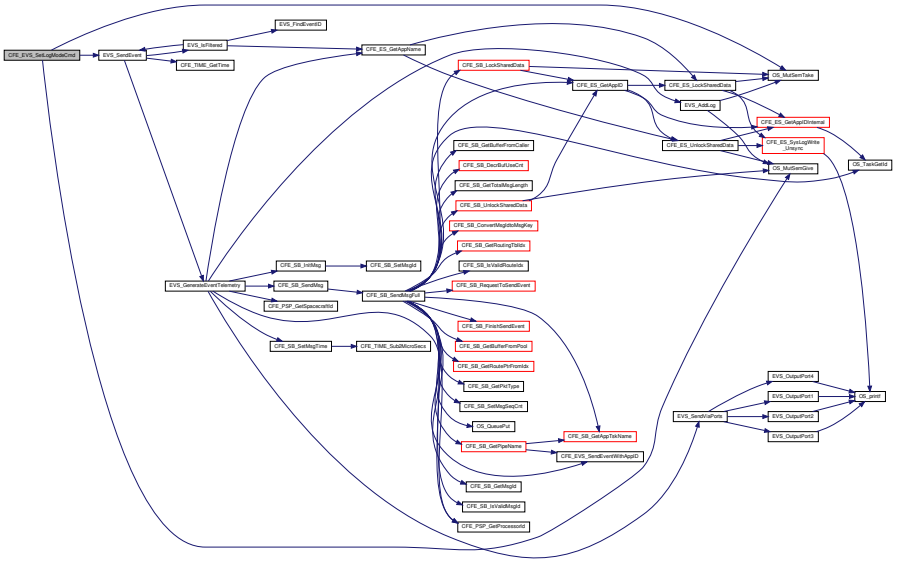
```
int32 CFE_EVS_SetLogModeCmd (
            const CFE_EVS_SetLogMode_t * data )
```

Definition at line 273 of file cfe_evs_log.c.

References CFE_EVS_ERR_LOGMODE_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_FUNCTION_DISABLED, CFE_EVS_GlobalData, CFE_EVS_INVALID_PARAMETER, CFE_EVS_LogMode_DISCARD, CFE_EVS_LOGMODE_EID, CFE_EVS_LogMode_OVERWRITE, CFE_EVS_NO_LOGSET_EID, CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_LogPtr, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogMode, CFE_EVS_SetLogMode_Payload_t::LogMode, OS_MutSemGive(), OS_MutSemTake(), CFE_EVS_SetLogMode_t::Payload, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.40.1.2 CFE_EVS_WriteLogDataFileCmd()

```
int32 CFE_EVS_WriteLogDataFileCmd (
            const CFE_EVS_WriteLogDataFile_t * data )
```

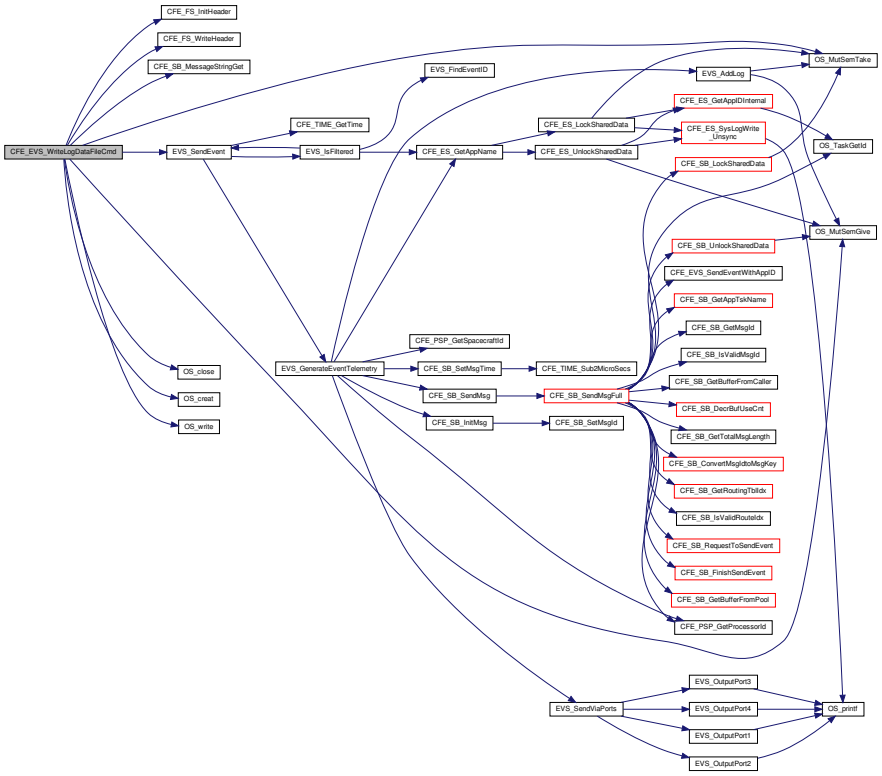
Definition at line 151 of file cfe_evs_log.c.

References CFE_EVS_ERR_CRLOGFILE_EID, CFE_EVS_ERR_WRLOGFILE_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_FILE_WRITE_ERROR, CFE_EVS_FUNCTION_DISABLED, CFE_EVS_GlobalData, CFE_EVS_NO_LOGWR_EID, CFE_EVS_WRLOG_EID, CFE_FS_InitHeader(), CFE_FS_SubType_EVS_EVENTLOG, CFE_FS_WriteHeader(), CFE_PLATFORM_EVS_DEFAULT_LOG_FILE, CFE_PLATFORM_EVS_LOG_MAX, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_LogPtr,

EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_Log_t::LogCount, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogEntry, CFE_EVS_LogFileCmd_Payload_t::LogFilename, CFE_EVS_Log_t::Next, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_MAX_PATH_LEN, OS_MutSemGive(), OS_MutSemTake(), OS_write(), OS_WRITE_ONLY, CFE_EVS_WriteLogDataFile_t::Payload, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.40.1.3 EVS_AddLog()

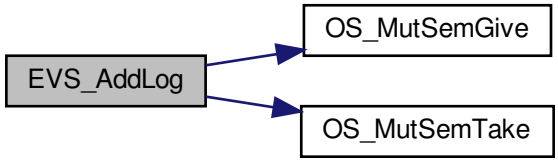
```
void EVS_AddLog (
            CFE_EVS_LongEventTlm_t * EVS_PktPtr )
```

Definition at line 54 of file cfe_evs_log.c.

References CFE_EVS_GlobalData, CFE_EVS_LogMode_DISCARD, CFE_PLATFORM_EVS_LOG_MAX, CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_Log_t::LogCount, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogEntry, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_Log_t::LogMode, CFE_EVS_Log_t::LogOverflowCounter, CFE_EVS_Log_t::Next, OS_MutSemGive(), OS_MutSemTake(), and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.40.1.4 EVS_ClearLog()

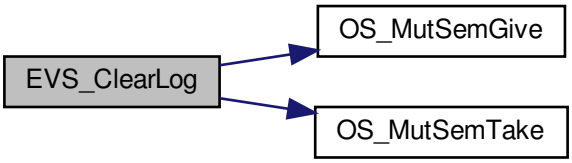
```
void EVS_ClearLog (
             void  )
```

Definition at line 119 of file cfe_evs_log.c.

References CFE_EVS_GlobalData, CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_Shared↩ DataMutexID, CFE_EVS_Log_t::LogCount, CFE_EVS_Log_t::LogEntry, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_↩ Log_t::LogOverflowCounter, CFE_EVS_Log_t::Next, OS_MutSemGive(), and OS_MutSemTake().

Referenced by CFE_EVS_ClearLogCmd(), and CFE_EVS_EarlyInit().

Here is the call graph for this function:



13.41 cfe/fsw/cfe-core/src/evs/cfe_evs_log.h File Reference

```
#include "cfe_evs_msg.h"
```

Functions

- void EVS_AddLog (CFE_EVS_LongEventTlm_t *EVS_PktPtr)
- void EVS_ClearLog (void)
- int32 CFE_EVS_WriteLogDataFileCmd (const CFE_EVS_WriteLogDataFile_t *data)
- int32 CFE_EVS_SetLogModeCmd (const CFE_EVS_SetLogMode_t *data)

13.41.1 Function Documentation

13.41.1.1 CFE_EVS_SetLogModeCmd()

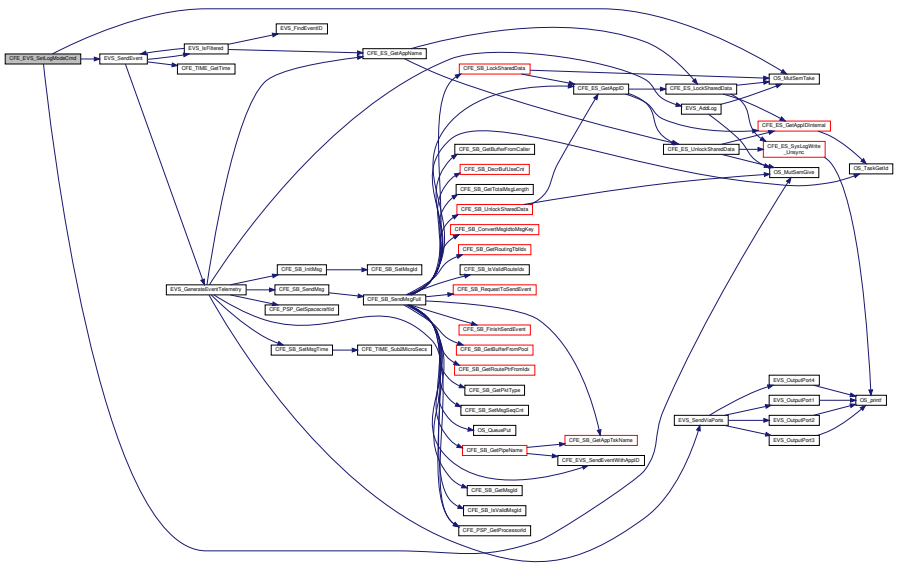
```
int32 CFE_EVS_SetLogModeCmd (
            const CFE_EVS_SetLogMode_t * data )
```

Definition at line 273 of file cfe_evs_log.c.

References CFE_EVS_ERR_LOGMODE_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_FUNCTION_DISABLED, CFE_EVS_GlobalData, CFE_EVS_INVALID_PARAMETER, CFE_EVS_LogMode_DISCARD, CFE_EVS_LOGMODE_EID, CFE_EVS_LogMode_OVERWRITE, CFE_EVS_NO_LOGSET_EID, CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_LogPtr, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogMode, CFE_EVS_SetLogMode_Payload_t::LogMode, OS_MutSemGive(), OS_MutSemTake(), CFE_EVS_SetLogMode_t::Payload, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.41.1.2 CFE_EVS_WriteLogDataFileCmd()

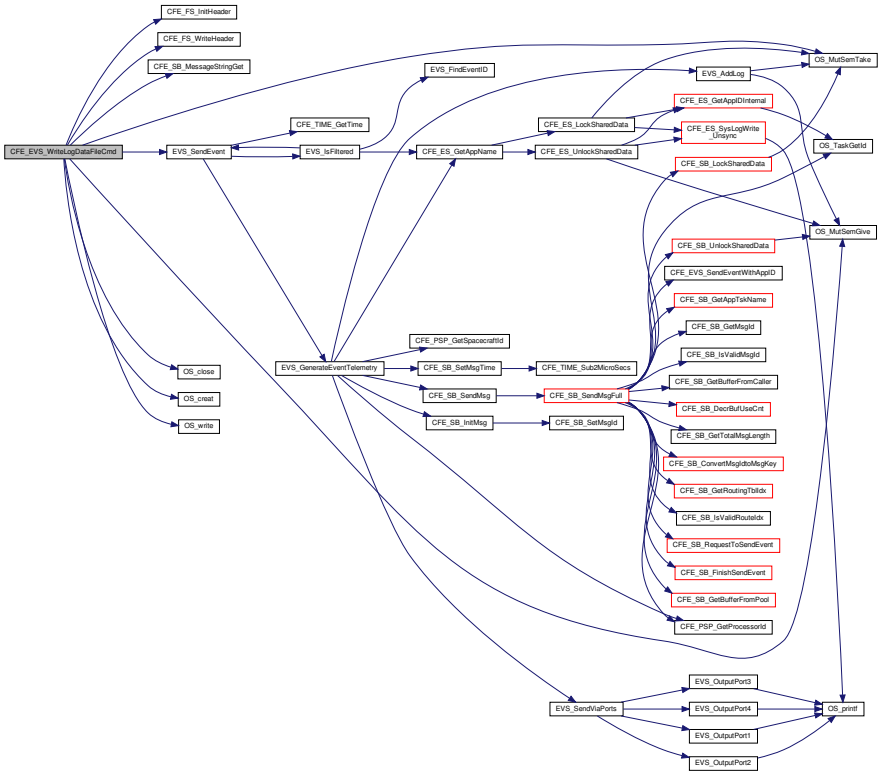
```
int32 CFE_EVS_WriteLogDataFileCmd (
            const CFE_EVS_WriteLogDataFile_t * data )
```

Definition at line 151 of file cfe_evs_log.c.

References CFE_EVS_ERR_CRLOGFILE_EID, CFE_EVS_ERR_WRLOGFILE_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_FILE_WRITE_ERROR, CFE_EVS_FUNCTION_DISABLED, CFE_EVS_GlobalData, CFE_EVS_NO_LOGWR_EID, CFE_EVS_WRLOG_EID, CFE_FS_InitHeader(), CFE_FS_SubType_EVS_EVENTLOG, CFE_FS_WriteHeader(), CFE_PLATFORM_EVS_DEFAULT_LOG_FILE, CFE_PLATFORM_EVS_LOG_MAX, CFE_SB_MessageStringGet(), CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_LogPtr, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_Log_t::LogCount, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogEntry, CFE_EVS_LogFileCmd_Payload_t::LogFilename, CFE_EVS_Log_t::Next, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_MAX_PATH_LEN, OS_MutSemGive(), OS_MutSemTake(), OS_write(), OS_WRITE_ONLY, CFE_EVS_WriteLogDataFile_t::Payload, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.41.1.3 EVS_AddLog()

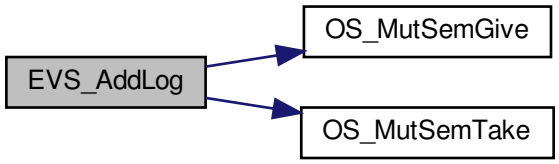
```
void EVS_AddLog (
            CFE_EVS_LongEventTlm_t * EVS_PktPtr )
```

Definition at line 54 of file cfe_evs_log.c.

References CFE_EVS_GlobalData, CFE_EVS_LogMode_DISCARD, CFE_PLATFORM_EVS_LOG_MAX, CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_Log_t::LogCount, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogEntry, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_Log_t::LogMode, CFE_EVS_Log_t::LogOverflowCounter, CFE_EVS_Log_t::Next, OS_MutSemGive(), OS_MutSemTake(), and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.41.1.4 EVS_ClearLog()

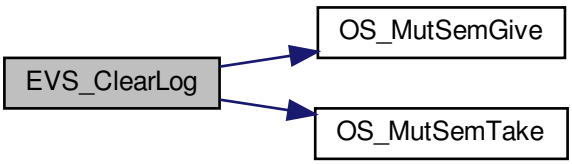
```
void EVS_ClearLog (
            void  )
```

Definition at line 119 of file cfe_evs_log.c.

References CFE_EVS_GlobalData, CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_Log_t::LogCount, CFE_EVS_Log_t::LogEntry, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_Log_t::LogOverflowCounter, CFE_EVS_Log_t::Next, OS_MutSemGive(), and OS_MutSemTake().

Referenced by CFE_EVS_ClearLogCmd(), and CFE_EVS_EarlyInit().

Here is the call graph for this function:



13.42 cfe/fsw/cfe-core/src/evs/cfe_evs_task.c File Reference

```
#include "cfe_evs_task.h"
#include "cfe_evs_log.h"
#include "cfe_evs_utils.h"
#include "cfe_evs.h"
#include <string.h>
#include "cfe_version.h"
#include "cfe_error.h"
#include "cfe_es.h"
#include "cfe_fs.h"
#include "cfe_psp.h"
#include "osapi.h"
#include "private/cfe_es_resetdata_typedef.h"
```

Functions

- void CFE_EVS_ProcessGroundCommand (CFE_SB_MsgPtr_t EVS_MsgPtr)
- bool CFE_EVS_VerifyCmdLength (CFE_SB_MsgPtr_t Msg, uint16 ExpectedLength)
- int32 CFE_EVS_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_EVS_CleanUpApp (uint32 AppID)

  Removes EVS resources associated with specified Application.
- void CFE_EVS_TaskMain (void)

  Entry Point for cFE Core Application.
- int32 CFE_EVS_TaskInit (void)
- void CFE_EVS_ProcessCommandPacket (CFE_SB_MsgPtr_t EVS_MsgPtr)
- int32 CFE_EVS_NoopCmd (const CFE_EVS_Noop_t *data)
- int32 CFE_EVS_ClearLogCmd (const CFE_EVS_ClearLog_t *data)
- int32 CFE_EVS_ReportHousekeepingCmd (const CCSDS_CommandPacket_t *data)
- int32 CFE_EVS_ResetCountersCmd (const CFE_EVS_ResetCounters_t *data)
- int32 CFE_EVS_SetFilterCmd (const CFE_EVS_SetFilter_t *data)
- int32 CFE_EVS_EnablePortsCmd (const CFE_EVS_EnablePorts_t *data)
- int32 CFE_EVS_DisablePortsCmd (const CFE_EVS_DisablePorts_t *data)
- int32 CFE_EVS_EnableEventTypeCmd (const CFE_EVS_EnableEventType_t *data)
- int32 CFE_EVS_DisableEventTypeCmd (const CFE_EVS_DisableEventType_t *data)
- int32 CFE_EVS_SetEventFormatModeCmd (const CFE_EVS_SetEventFormatMode_t *data)
- int32 CFE_EVS_EnableAppEventTypeCmd (const CFE_EVS_EnableAppEventType_t *data)
- int32 CFE_EVS_DisableAppEventTypeCmd (const CFE_EVS_DisableAppEventType_t *data)
- int32 CFE_EVS_EnableAppEventsCmd (const CFE_EVS_EnableAppEvents_t *data)
- int32 CFE_EVS_DisableAppEventsCmd (const CFE_EVS_DisableAppEvents_t *data)
- int32 CFE_EVS_ResetAppCounterCmd (const CFE_EVS_ResetAppCounter_t *data)
- int32 CFE_EVS_ResetFilterCmd (const CFE_EVS_ResetFilter_t *data)
- int32 CFE_EVS_ResetAllFiltersCmd (const CFE_EVS_ResetAllFilters_t *data)
- int32 CFE_EVS_AddEventFilterCmd (const CFE_EVS_AddEventFilter_t *data)
- int32 CFE_EVS_DeleteEventFilterCmd (const CFE_EVS_DeleteEventFilter_t *data)
- int32 CFE_EVS_WriteAppDataFileCmd (const CFE_EVS_WriteAppDataFile_t *data)

Variables

- CFE_EVS_GlobalData_t CFE_EVS_GlobalData

13.42.1 Function Documentation

13.42.1.1 CFE_EVS_AddEventFilterCmd()

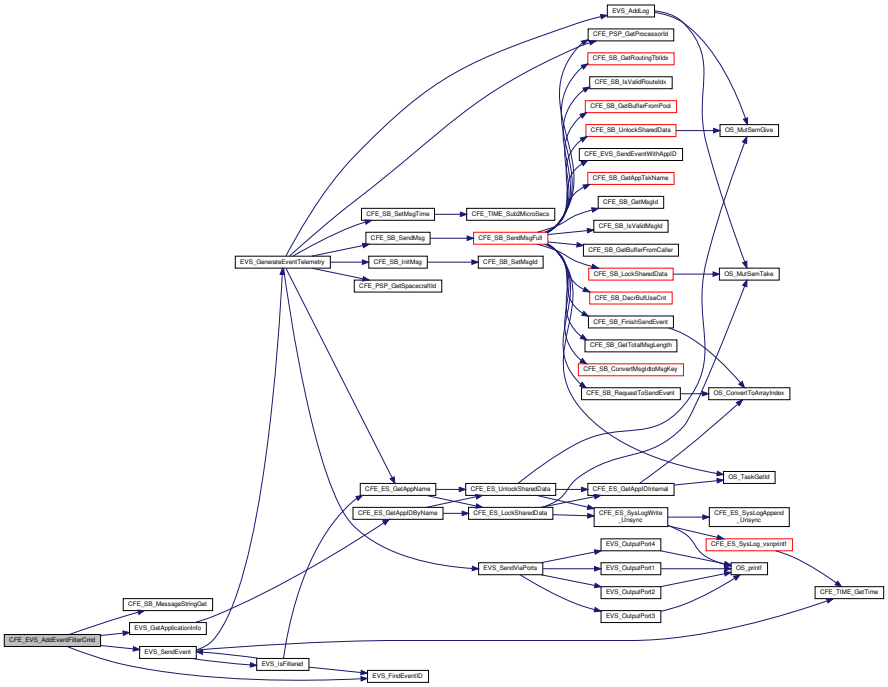
```
int32 CFE_EVS_AddEventFilterCmd (
            const CFE_EVS_AddEventFilter_t * data )
```

Definition at line 1555 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::AppName, EVS_AppData_t::BinFilters, CFE_EVS_ADD_EVENT_FILTER_CC, CFE_EVS_ADDFILTER_EID, CFE_EVS_APP_FILTER_OVERLOAD, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_MAXREGSFILTER_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_FILTERED_EID, CFE_EVS_EVT_NOT_REGISTERED, CFE_EVS_FREE_SLOT, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_BinFilter_t::Count, EVS_BinFilter_t::EventID, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::EventID, EVS_FindEventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), EVS_BinFilter_t::Mask, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::Mask, NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameEventIDMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.2 CFE_EVS_CleanUpApp()

```
int32 CFE_EVS_CleanUpApp (
            uint32 AppId )
```

Description

This function is called by cFE Executive Services to cleanup after an Application has been terminated. It frees resources that have been allocated to the specified Application.

Definition at line 184 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_PLATFORM_ES_MAX_A↩ PPLICATIONS, CFE_SUCCESS, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_ES_CleanUpApp().

13.42.1.3 CFE_EVS_ClearLogCmd()

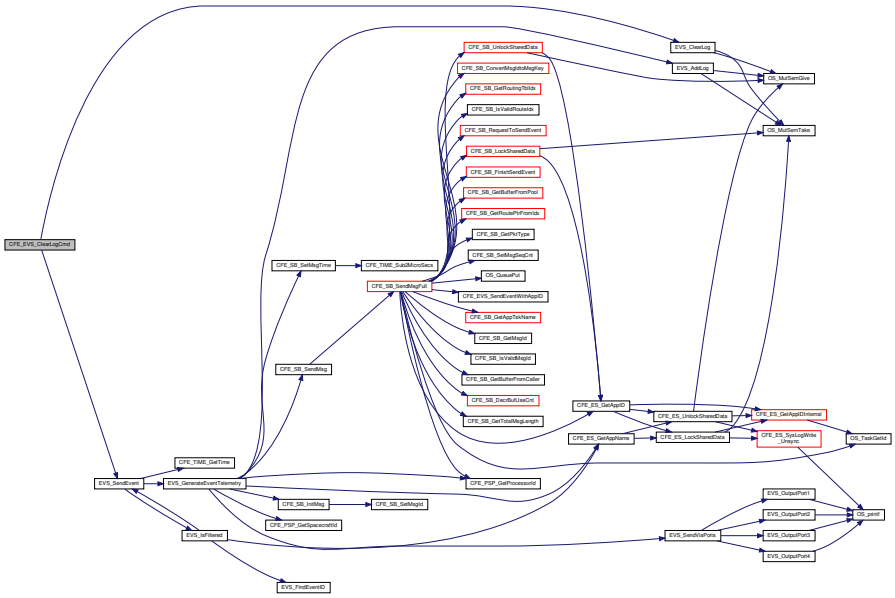
```
int32 CFE_EVS_ClearLogCmd (
            const CFE_EVS_ClearLog_t * data )
```

Definition at line 656 of file cfe_evs_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_FUNCTION_DISABLED, CFE_EVS_NO_LOGCLR_E↩ ID, CFE_SUCCESS, EVS_ClearLog(), EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_↩ HousekeepingTlm_Payload_t::LogEnabled, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.4 CFE_EVS_DeleteEventFilterCmd()

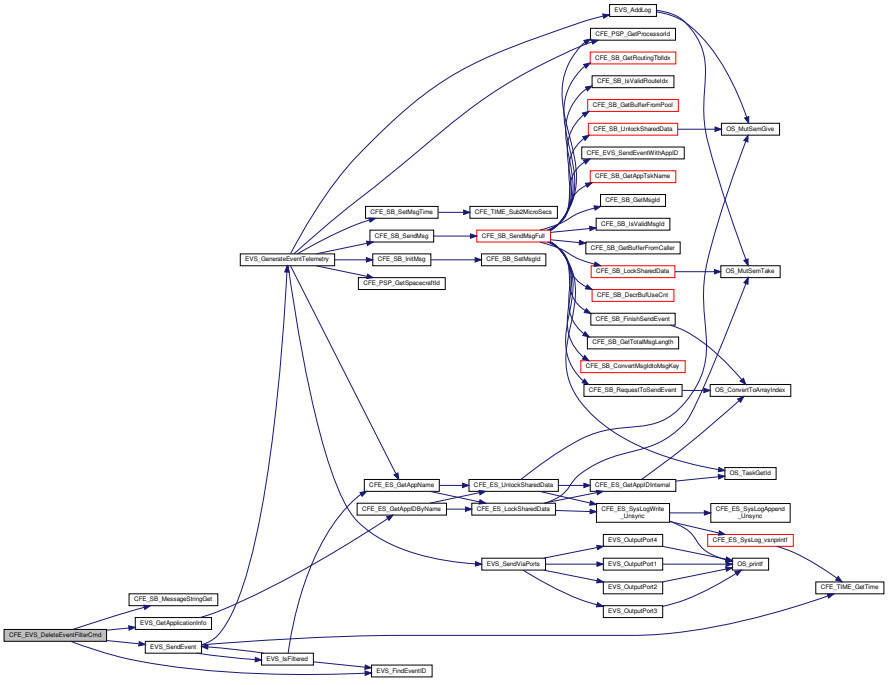
```
int32 CFE_EVS_DeleteEventFilterCmd (
            const CFE_EVS_DeleteEventFilter_t * data )
```

Definition at line 1650 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDCmd_Payload_t::AppName, EVS_App↩
Data_t::BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_DELETE_↩
EVENT_FILTER_CC, CFE_EVS_DELFILTER_EID, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_EVTIDN↩
OREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_Event↩
Type_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_NOT_REGISTERED, CFE_EVS_FREE_SLOT, CF↩
E_EVS_NO_MASK, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_BinFilter_t::↩
Count, EVS_BinFilter_t::EventID, CFE_EVS_AppNameEventIDCmd_Payload_t::EventID, EVS_FindEventID(), EVS_↩
GetApplicationInfo(), EVS_SendEvent(), EVS_BinFilter_t::Mask, NULL, OS_MAX_API_NAME, and CFE_EVS_App↩
NameEventIDCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.5 CFE_EVS_DisableAppEventsCmd()

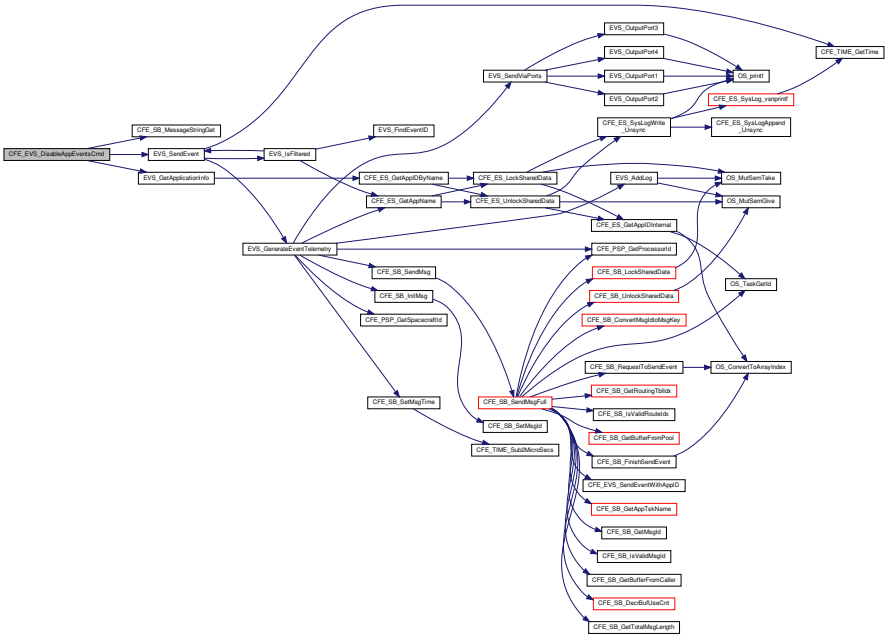
```
int32 CFE_EVS_DisableAppEventsCmd (
            const CFE_EVS_DisableAppEvents_t * data )
```

Definition at line 1299 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_DISAPPEVT_EID, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.6 CFE_EVS_DisableAppEventTypeCmd()

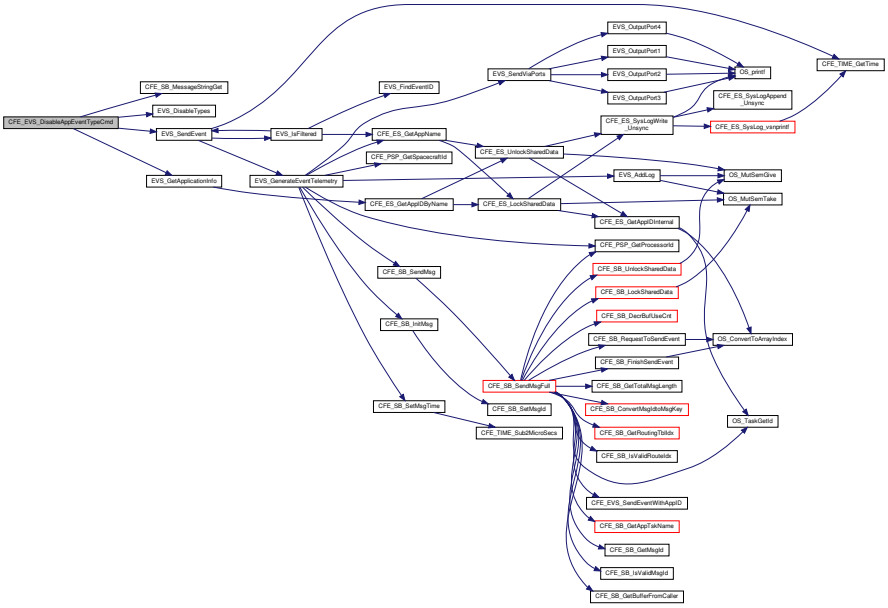
```
int32 CFE_EVS_DisableAppEventTypeCmd (
            const CFE_EVS_DisableAppEventType_t * data )
```

Definition at line 1168 of file cfe_evs_task.c.

References CFE_EVS_AppNameBitMaskCmd_Payload_t::AppName, CFE_EVS_AppNameBitMaskCmd_Payload_t::BitMask, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISAPPEVENTTYPE_EID, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_DisableTypes(), EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameBitMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.7 CFE_EVS_DisableEventTypeCmd()

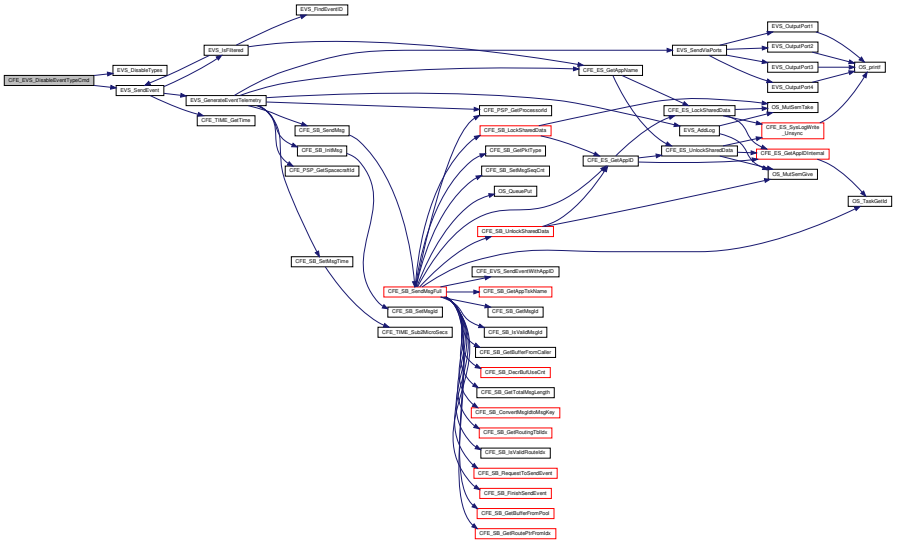
```
int32 CFE_EVS_DisableEventTypeCmd (
            const CFE_EVS_DisableEventType_t * data )
```

Definition at line 1007 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_DISEVTTYPE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, EVS_DisableTypes(), EVS_SendEvent(), CFE_EVS_BitMaskCmd_t::Payload, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.8 CFE_EVS_DisablePortsCmd()

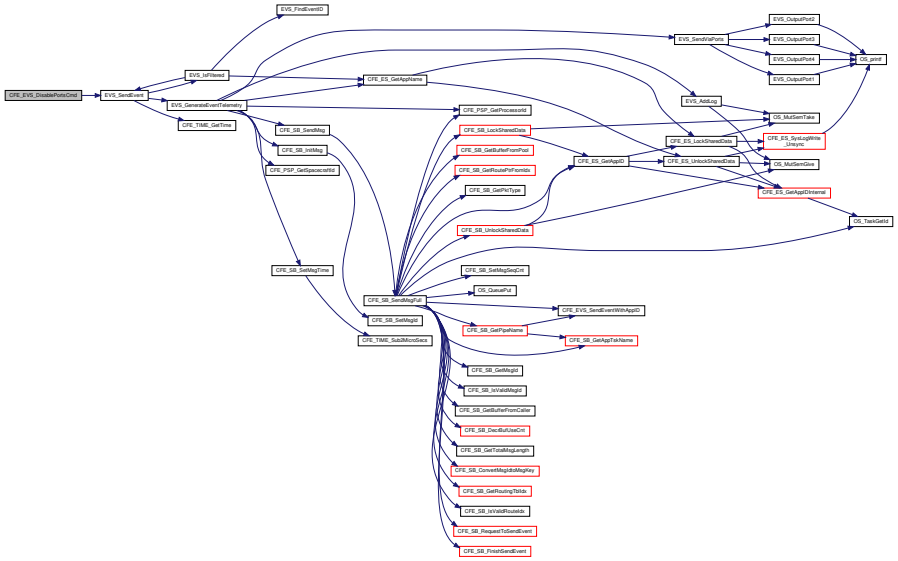
```
int32 CFE_EVS_DisablePortsCmd (
            const CFE_EVS_DisablePorts_t * data )
```

Definition at line 901 of file cfe_evs_task.c.

References CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_DISABLE_PORTS_CC, CFE_EVS_DISPOR↩T_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT, C↩FE_EVS_PORT4_BIT, CFE_SUCCESS, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_↩HousekeepingTlm_Payload_t::OutputPort, CFE_EVS_BitMaskCmd_t::Payload, and CFE_EVS_HousekeepingTlm_t↩::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.9 CFE_EVS_EarlyInit()

```
int32 CFE_EVS_EarlyInit (
            void  )
```

cFE Core task early init prototypes

Description

Initializes the cFE core module API Library

Assumptions, External Events, and Notes:

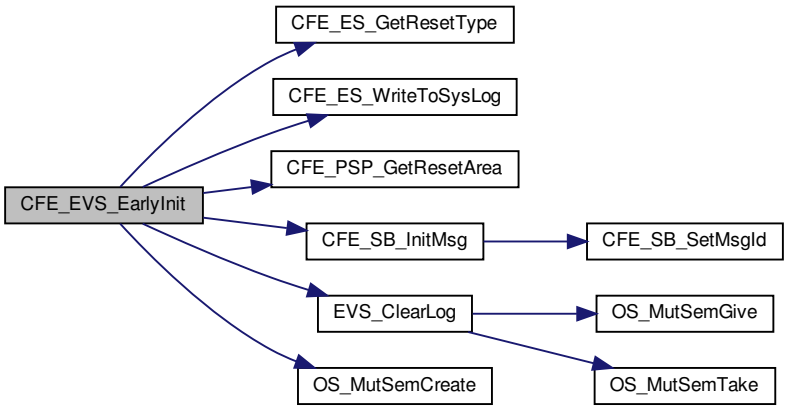
1. This function MUST be called before any module API's are called.

Definition at line 74 of file cfe_evs_task.c.

References CFE_ES_GetResetType(), CFE_ES_WriteToSysLog(), CFE_EVS_HK_TLM_MID, CFE_EVS_LogMode↩
_DISCARD, CFE_EVS_LogMode_OVERWRITE, CFE_EVS_RESET_AREA_POINTER, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_DEFAULT_LOG_MODE, CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE, C↩
FE_PLATFORM_EVS_LOG_MAX, CFE_PLATFORM_EVS_PORT_DEFAULT, CFE_PSP_GetResetArea(), CFE_P↩
SP_RST_TYPE_POWERON, CFE_PSP_SUCCESS, CFE_SB_InitMsg(), CFE_SUCCESS, CFE_EVS_GlobalData↩
_t::EVS_AppID, EVS_ClearLog(), CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_Shared↩
DataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_Log_t::LogCount, CFE_EVS_HousekeepingTlm_↩
Payload_t::LogEnabled, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_HousekeepingTlm_Payload_t::LogFullFlag, CFE↩
_EVS_Log_t::LogMode, CFE_EVS_HousekeepingTlm_Payload_t::LogMode, CFE_EVS_Log_t::LogOverflowCounter,

CFE_EVS_HousekeepingTlm_Payload_t::MessageFormatMode, CFE_EVS_Log_t::Next, NULL, OS_MutSemCreate(), OS_SUCCESS, CFE_EVS_HousekeepingTlm_Payload_t::OutputPort, and CFE_EVS_HousekeepingTlm_t::Payload.

Here is the call graph for this function:



13.42.1.10 CFE_EVS_EnableAppEventsCmd()

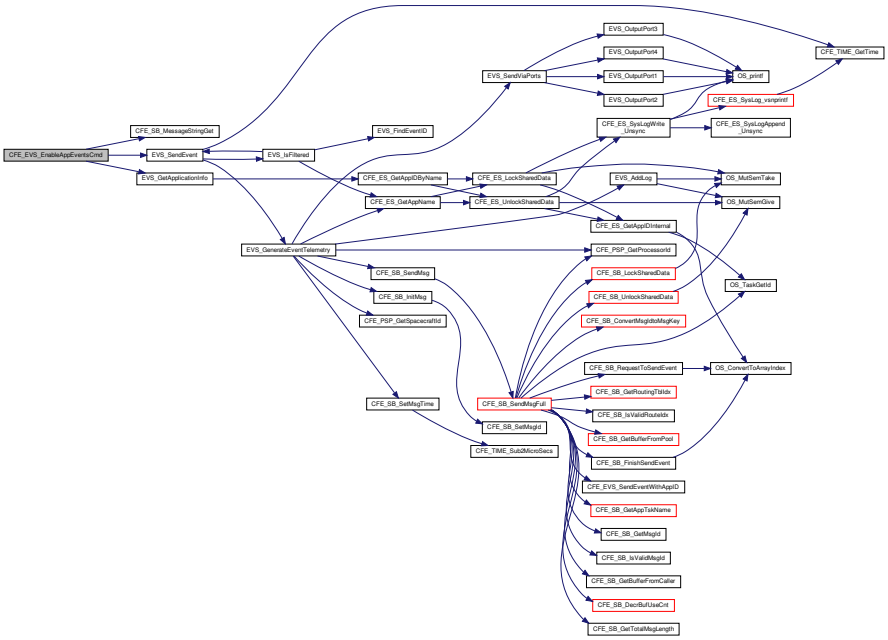
```
int32 CFE_EVS_EnableAppEventsCmd (
            const CFE_EVS_EnableAppEvents_t * data )
```

Definition at line 1241 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ENAAPPEVT_EID, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.11 CFE_EVS_EnableAppEventTypeCmd()

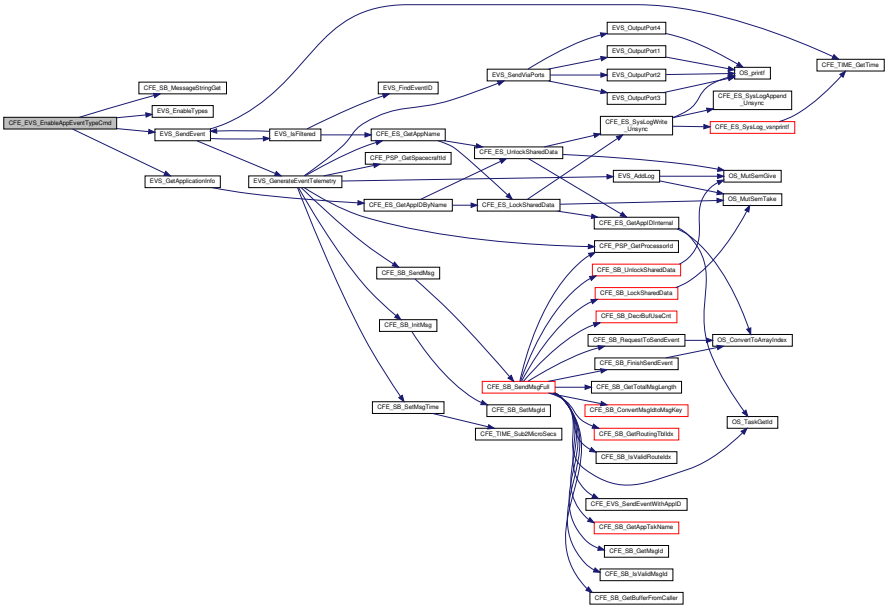
```
int32 CFE_EVS_EnableAppEventTypeCmd (
            const CFE_EVS_EnableAppEventType_t * data )
```

Definition at line 1094 of file cfe_evs_task.c.

References CFE_EVS_AppNameBitMaskCmd_Payload_t::AppName, CFE_EVS_AppNameBitMaskCmd_Payload_t::BitMask, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ENAAPPEVTTYPE_EID, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_EnableTypes(), EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameBitMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.12 CFE_EVS_EnableEventTypeCmd()

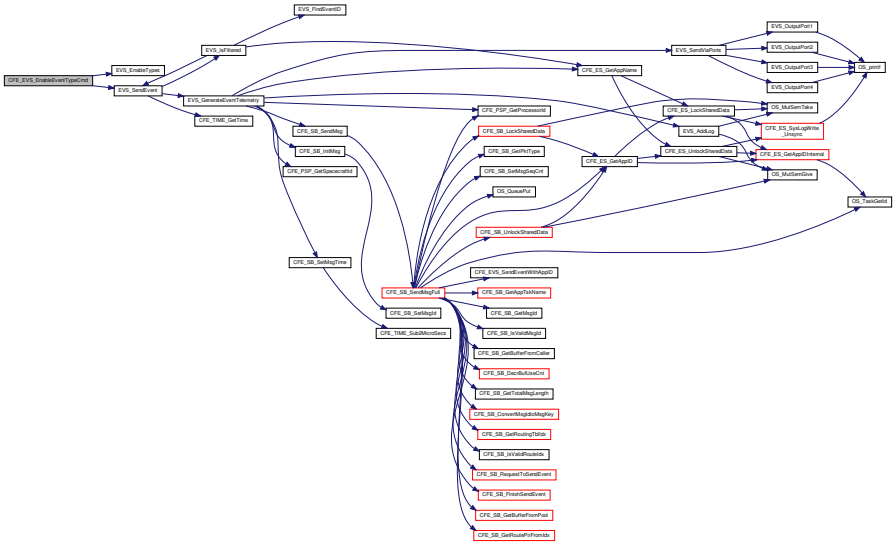
```
int32 CFE_EVS_EnableEventTypeCmd (
            const CFE_EVS_EnableEventType_t * data )
```

Definition at line 959 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_ENABLE↩
_EVENT_TYPE_CC, CFE_EVS_ENAEVTTYPE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_Event↩
Type_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_PLATFORM_ES_MAX_↩
APPLICATIONS, CFE_SUCCESS, EVS_EnableTypes(), EVS_SendEvent(), CFE_EVS_BitMaskCmd_t::Payload, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.13 CFE_EVS_EnablePortsCmd()

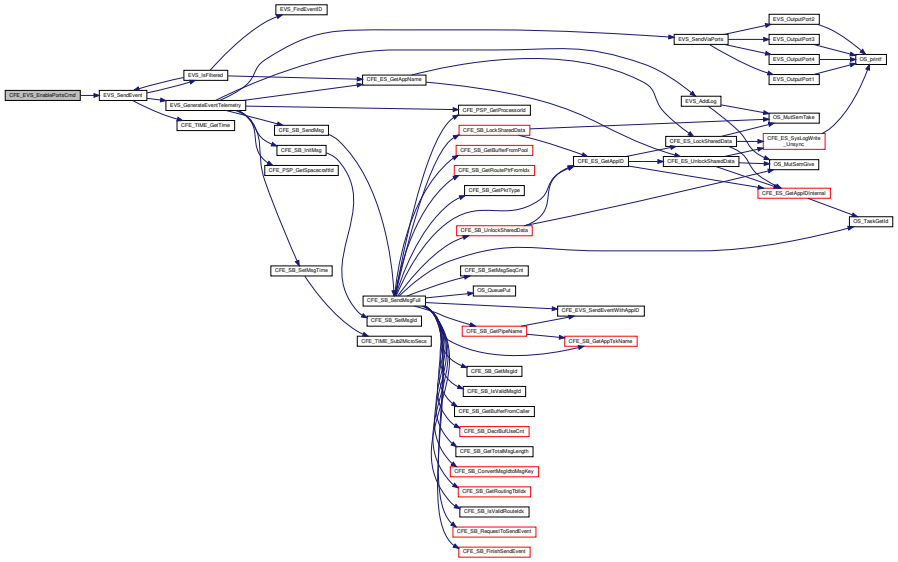
```
int32 CFE_EVS_EnablePortsCmd (
            const CFE_EVS_EnablePorts_t * data )
```

Definition at line 845 of file cfe_evs_task.c.

References CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_ENABLE_PORTS_CC, CFE_EVS_ENAPO↩RT_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERR↩OR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT, CFE_EVS_PORT4_BIT, CFE_SUCCESS, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_↩HousekeepingTlm_Payload_t::OutputPort, CFE_EVS_BitMaskCmd_t::Payload, and CFE_EVS_HousekeepingTlm_t↩::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.14 CFE_EVS_NoopCmd()

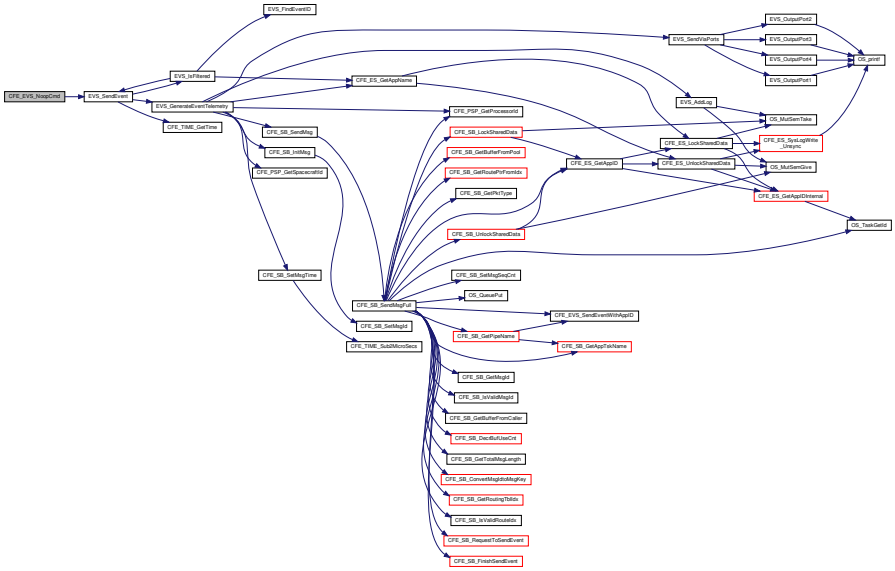
```
int32 CFE_EVS_NoopCmd (
            const CFE_EVS_Noop_t * data )
```

Definition at line 639 of file cfe_evs_task.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_NOOP_EID, CFE_MAJOR_VERSION, CFE_MINOR↩
_VERSION, CFE_MISSION_REV, CFE_REVISION, CFE_SUCCESS, and EVS_SendEvent().

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.15 CFE_EVS_ProcessCommandPacket()

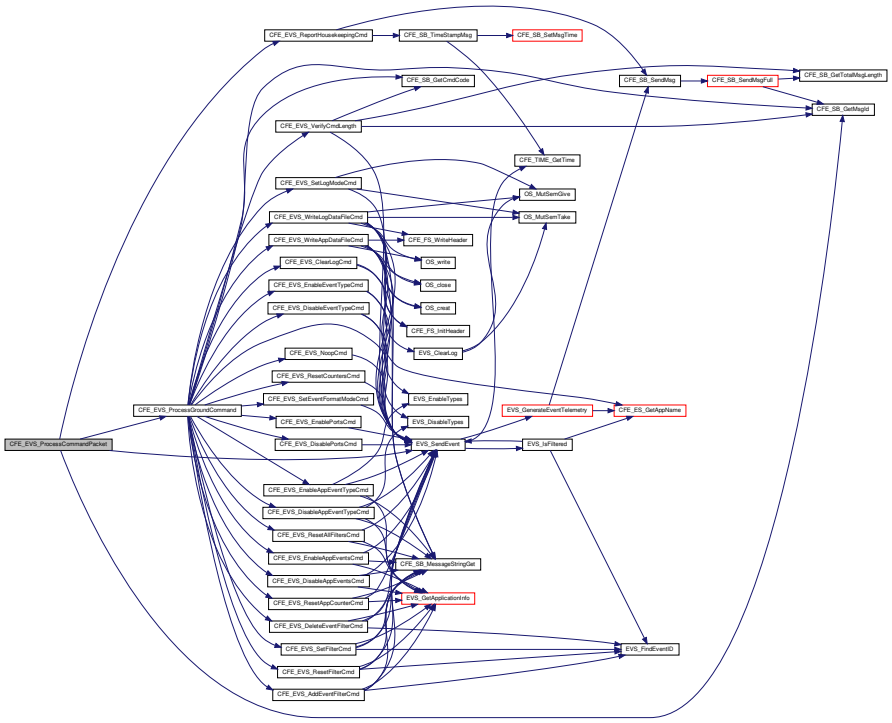
```
void CFE_EVS_ProcessCommandPacket (
            CFE_SB_MsgPtr_t EVS_MsgPtr )
```

Definition at line 355 of file cfe_evs_task.c.

References CFE_EVS_CMD_MID, CFE_EVS_ERR_MSGID_EID, CFE_EVS_EventType_ERROR, CFE_EVS_↩ ProcessGroundCommand(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_SEND_HK_MID, CFE_SB_Get↩ MsgId(), CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter, EVS_SendEvent(), CFE_EVS_Global↩ Data_t::EVS_TlmPkt, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_TaskMain().

Here is the call graph for this function:



13.42.1.16 CFE_EVS_ProcessGroundCommand()

```
void CFE_EVS_ProcessGroundCommand (
            CFE_SB_MsgPtr_t EVS_MsgPtr )
```

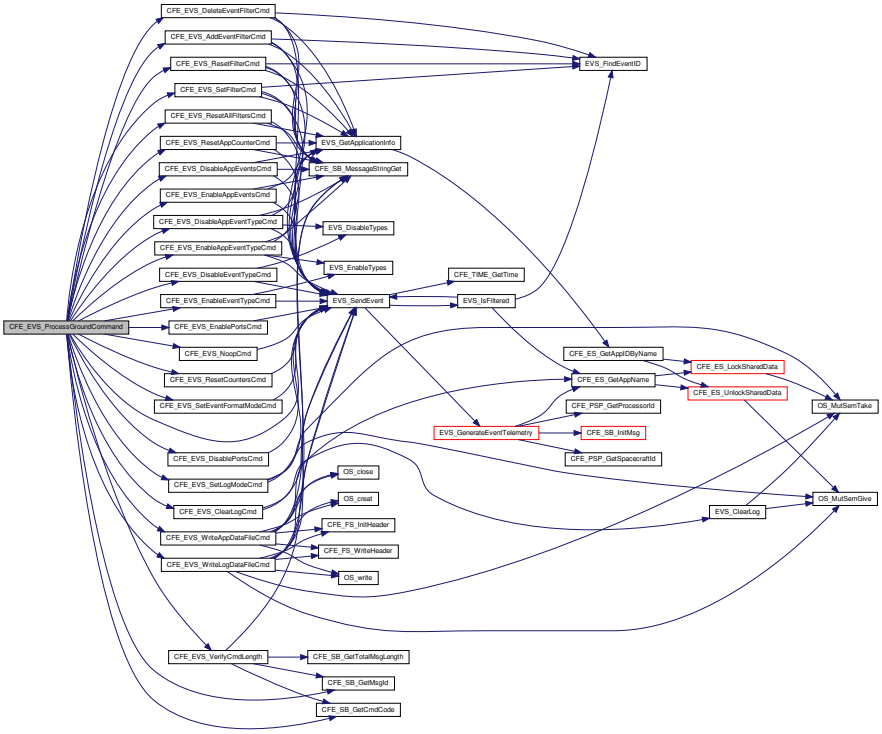
Definition at line 395 of file cfe_evs_task.c.

References CFE_EVS_ADD_EVENT_FILTER_CC, CFE_EVS_AddEventFilterCmd(), CFE_EVS_CLEAR_LOG_CC, CFE_EVS_ClearLogCmd(), CFE_EVS_DELETE_EVENT_FILTER_CC, CFE_EVS_DeleteEventFilterCmd(), CFE↩ _EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_EVE↩ NT_TYPE_CC, CFE_EVS_DISABLE_PORTS_CC, CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableApp↩ EventTypeCmd(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_ENABLE_APP_↩ EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EV↩ S_ENABLE_PORTS_CC, CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS↩ _EnableEventTypeCmd(), CFE_EVS_EnablePortsCmd(), CFE_EVS_ERR_CC_EID, CFE_EVS_EventType_ERROR, CFE_EVS_NOOP_CC, CFE_EVS_NoopCmd(), CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_RESET_AP↩ P_COUNTER_CC, CFE_EVS_RESET_COUNTERS_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_ResetAll↩ FiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetCountersCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SET_EVENT_FORMAT_MODE_CC, CFE_EVS_SET_FILTER_CC, CFE_EVS_SET_LOG_MODE_CC, CFE_EVS_SetEventFormatModeCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_SetLogModeCmd(), CFE_EVS_↩ VerifyCmdLength(), CFE_EVS_WRITE_APP_DATA_FILE_CC, CFE_EVS_WRITE_LOG_DATA_FILE_CC, CFE_↩ EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(),

CFE_STATUS_BAD_COMMAND_CODE, CFE_STATUS_WRONG_MSG_LENGTH, CFE_SUCCESS, CFE_EVS↩ _HousekeepingTlm_Payload_t::CommandCounter, CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessCommandPacket().

Here is the call graph for this function:



13.42.1.17 CFE_EVS_ReportHousekeepingCmd()

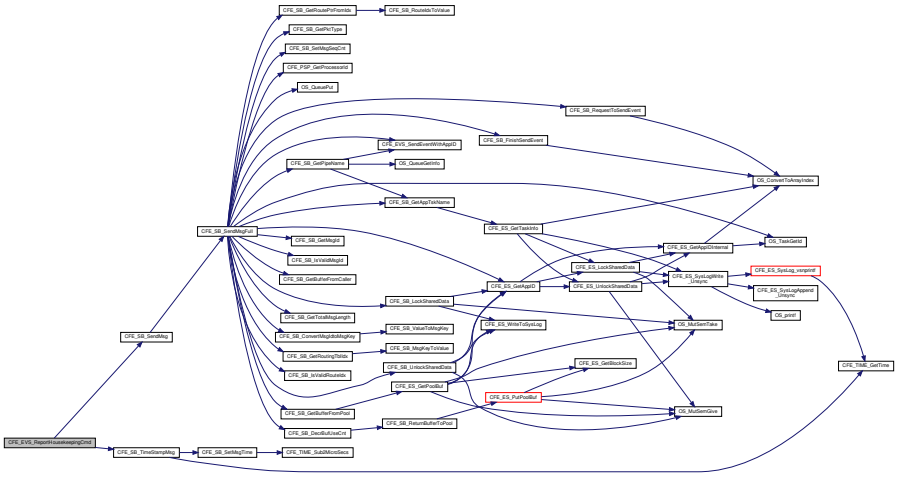
```
int32 CFE_EVS_ReportHousekeepingCmd (
            const CCSDS_CommandPacket_t * data )
```

Definition at line 684 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_HousekeepingTlm_Payload↩ _t::AppData, CFE_EVS_AppTlmData_t::AppEnableStatus, CFE_EVS_AppTlmData_t::AppID, CFE_EVS_AppTlm↩ Data_t::AppMessageSentCounter, CFE_MISSION_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_MAX_APPLI↩ CATIONS, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_STATUS_NO_COUNTER_INCREMENT, EVS_↩ AppData_t::EventCount, CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_↩ HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_HousekeepingTlm_Payload_↩ t::LogFullFlag, CFE_EVS_Log_t::LogMode, CFE_EVS_HousekeepingTlm_Payload_t::LogMode, CFE_EVS_Log_t::↩ LogOverflowCounter, CFE_EVS_HousekeepingTlm_Payload_t::LogOverflowCounter, CFE_EVS_HousekeepingTlm↩ _t::Payload, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_ProcessCommandPacket().

Here is the call graph for this function:



13.42.1.18 CFE_EVS_ResetAllFiltersCmd()

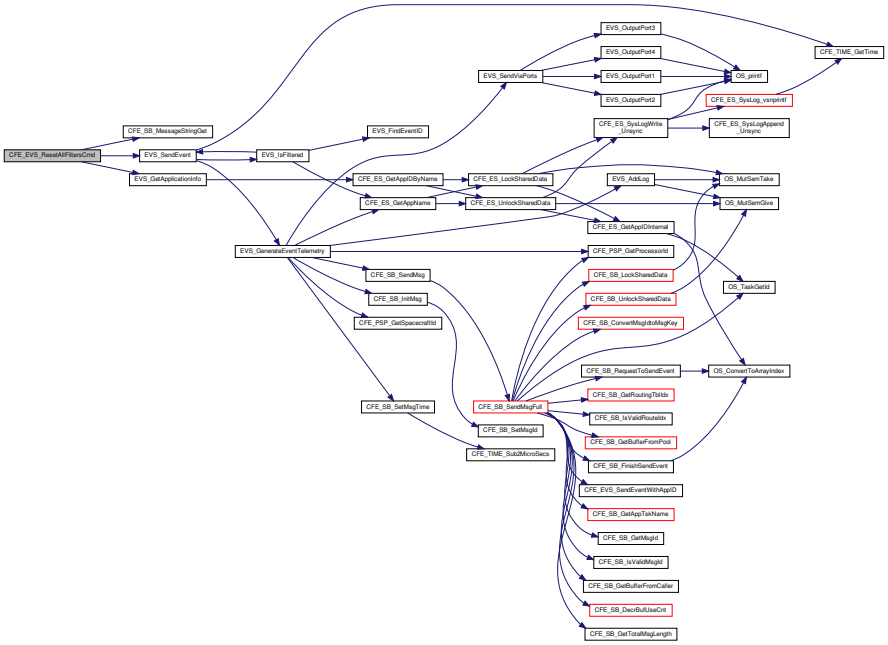
```
int32 CFE_EVS_ResetAllFiltersCmd (
            const CFE_EVS_ResetAllFilters_t * data )
```

Definition at line 1493 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, EVS_AppData_t::↩
BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_APPNORE↩
GS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType↩
_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_RSTALLFILTER_EID, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SB_MessageStringGet(), CFE↩
_SUCCESS, EVS_BinFilter_t::Count, EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.19 CFE_EVS_ResetAppCounterCmd()

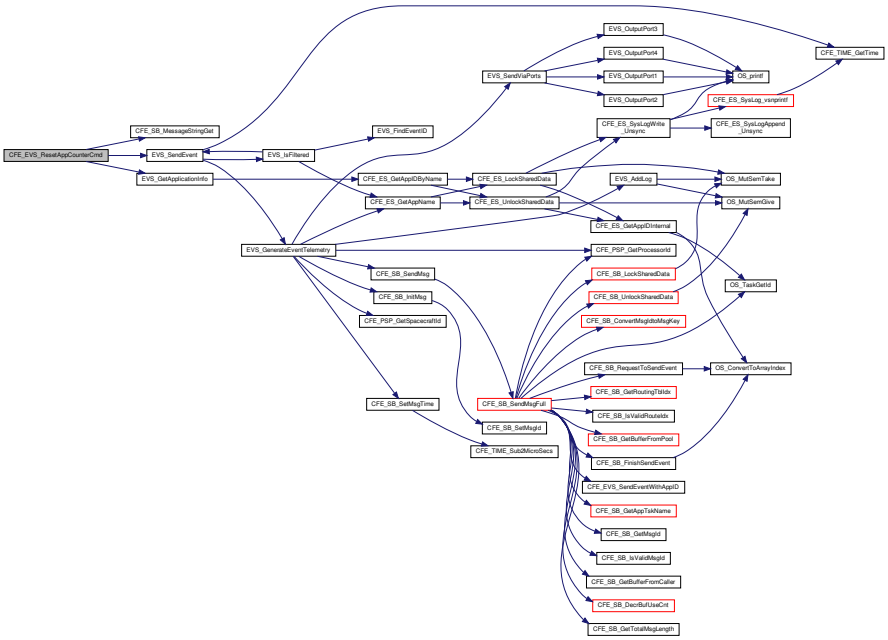
```
int32 CFE_EVS_ResetAppCounterCmd (
            const CFE_EVS_ResetAppCounter_t * data )
```

Definition at line 1358 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, CFE_EVS_APP_↩ ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_I↩ LLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_Event↩ Type_ERROR, CFE_EVS_RESET_APP_COUNTER_CC, CFE_EVS_RSTEVTCNT_EID, CFE_EVS_UNDEF_APP↩ ID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_AppData_t::EventCount, EVS_GetApplicationInfo(), EVS↩ _SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.20 CFE_EVS_ResetCountersCmd()

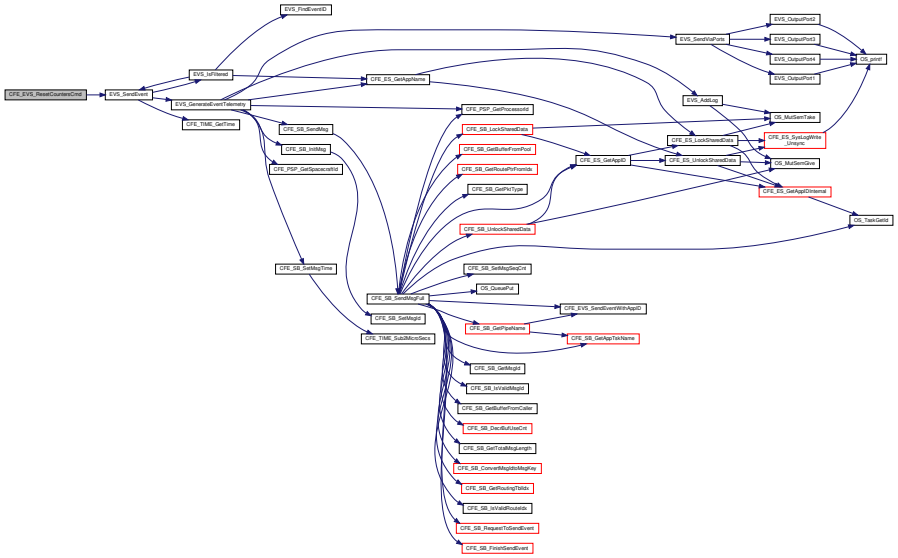
```
int32 CFE_EVS_ResetCountersCmd (
            const CFE_EVS_ResetCounters_t * data )
```

Definition at line 736 of file cfe_evs_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_RSTCNT_EID, CFE_STATUS_NO_COUNTER_INCREMENT, CFE_EVS_HousekeepingTlm_Payload_t::CommandCounter, CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_HousekeepingTlm_Payload_t::MessageSendCounter, CFE_EVS_HousekeepingTlm_Payload_t::MessageTruncCounter, CFE_EVS_HousekeepingTlm_Payload_t::Payload, and CFE_EVS_HousekeepingTlm_Payload_t::UnregisteredAppCounter.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.21 CFE_EVS_ResetFilterCmd()

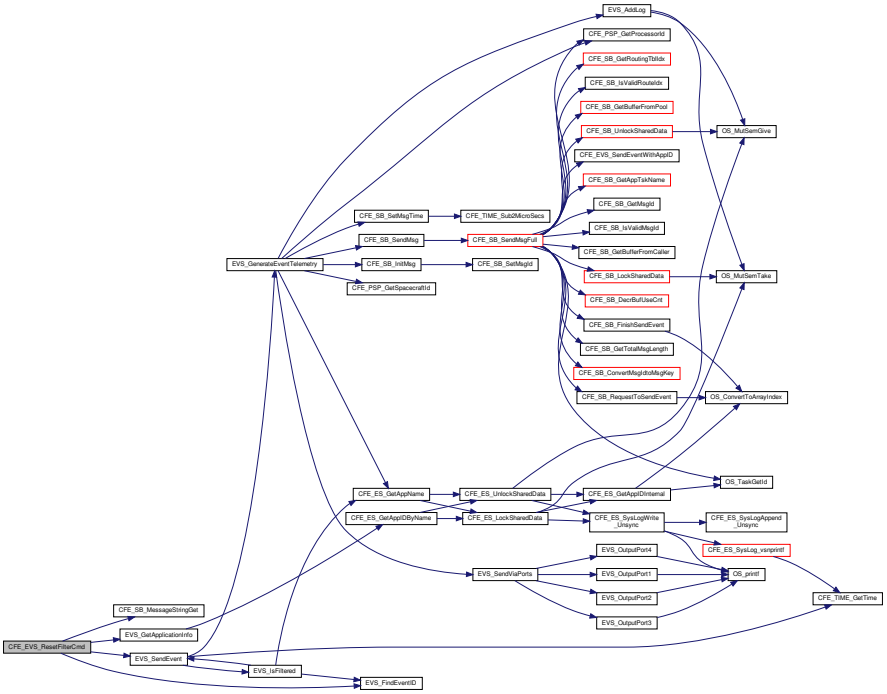
```
int32 CFE_EVS_ResetFilterCmd (
            const CFE_EVS_ResetFilter_t * data )
```

Definition at line 1417 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDCmd_Payload_t::AppName, EVS_App↩Data_t::BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_AP↩PNOREGS_EID, CFE_EVS_ERR_EVTIDNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR↩_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_NOT_R↩EGISTERED, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RSTFILTER_EID, CFE_EVS_UNDEF_APPID, CFE_SB_↩MessageStringGet(), CFE_SUCCESS, EVS_BinFilter_t::Count, CFE_EVS_AppNameEventIDCmd_Payload_t::Event↩ID, EVS_FindEventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS↩_AppNameEventIDCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.22 CFE_EVS_SetEventFormatModeCmd()

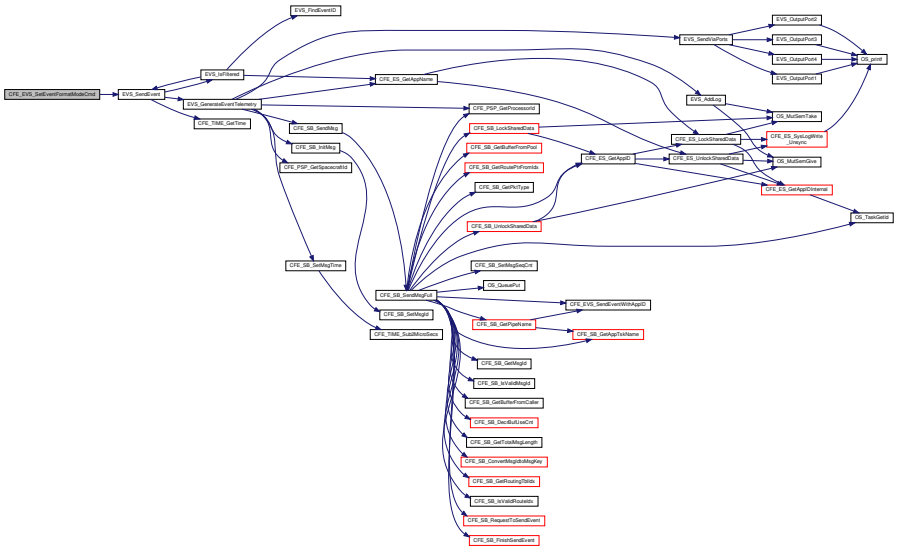
```
int32 CFE_EVS_SetEventFormatModeCmd (
            const CFE_EVS_SetEventFormatMode_t * data )
```

Definition at line 1056 of file cfe_evs_task.c.

References CFE_EVS_ERR_ILLEGALFMTMOD_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERR↩OR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_MsgFormat_LONG, CFE_EVS_MsgFormat_SHORT, CFE_EV↩S_SETEVTFMTMOD_EID, CFE_SUCCESS, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS↩_HousekeepingTlm_Payload_t::MessageFormatMode, CFE_EVS_SetEventFormatMode_Payload_t::MsgFormat, C↩FE_EVS_SetEventFormatMode_t::Payload, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.23 CFE_EVS_SetFilterCmd()

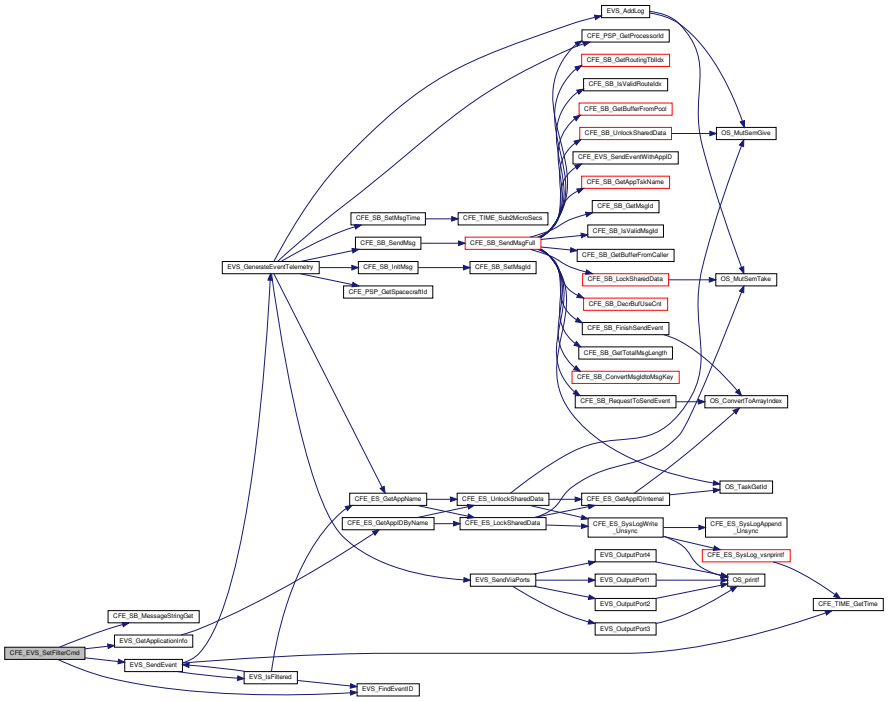
```
int32 CFE_EVS_SetFilterCmd (
            const CFE_EVS_SetFilter_t * data )
```

Definition at line 766 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::AppName, EVS↩
_AppData_t::BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR↩
_APPNOREGS_EID, CFE_EVS_ERR_EVTIDNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_E↩
RR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_NOT↩
_REGISTERED, CFE_EVS_SET_FILTER_CC, CFE_EVS_SETFILTERMSK_EID, CFE_EVS_UNDEF_APPID, CFE↩
_SB_MessageStringGet(), CFE_SUCCESS, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::EventID, EVS_Find↩
EventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), EVS_BinFilter_t::Mask, CFE_EVS_AppNameEventIDMask↩
Cmd_Payload_t::Mask, NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameEventIDMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.24 CFE_EVS_TaskInit()

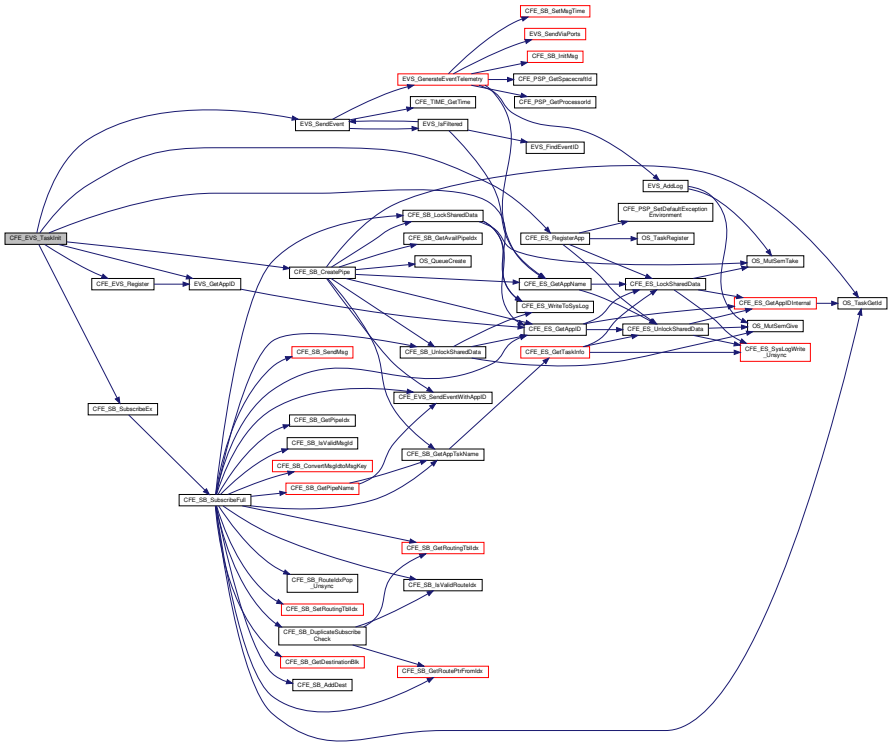
```
int32 CFE_EVS_TaskInit (
            void  )
```

Definition at line 279 of file cfe_evs_task.c.

References CFE_ES_RegisterApp(), CFE_ES_WriteToSysLog(), CFE_EVS_CMD_MID, CFE_EVS_EventFilter_B↩INARY, CFE_EVS_EventType_INFORMATION, CFE_EVS_MSG_LIMIT, CFE_EVS_PIPE_DEPTH, CFE_EVS_PI↩PE_NAME, CFE_EVS_Register(), CFE_EVS_SEND_HK_MID, CFE_EVS_STARTUP_EID, CFE_MAJOR_VERSI↩ON, CFE_MINOR_VERSION, CFE_MISSION_REV, CFE_REVISION, CFE_SB_CreatePipe(), CFE_SB_Default_Qos, CFE_SB_SubscribeEx(), CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_AppID, CFE_EVS_GlobalData_t::EVS_↩CommandPipe, EVS_GetAppID(), EVS_SendEvent(), and NULL.

Referenced by CFE_EVS_TaskMain().

Here is the call graph for this function:



13.42.1.25 CFE_EVS_TaskMain()

```
void CFE_EVS_TaskMain (
            void  )
```

Description

This is the entry point to the cFE EVS Core Application.

Assumptions, External Events, and Notes:

None

Return values

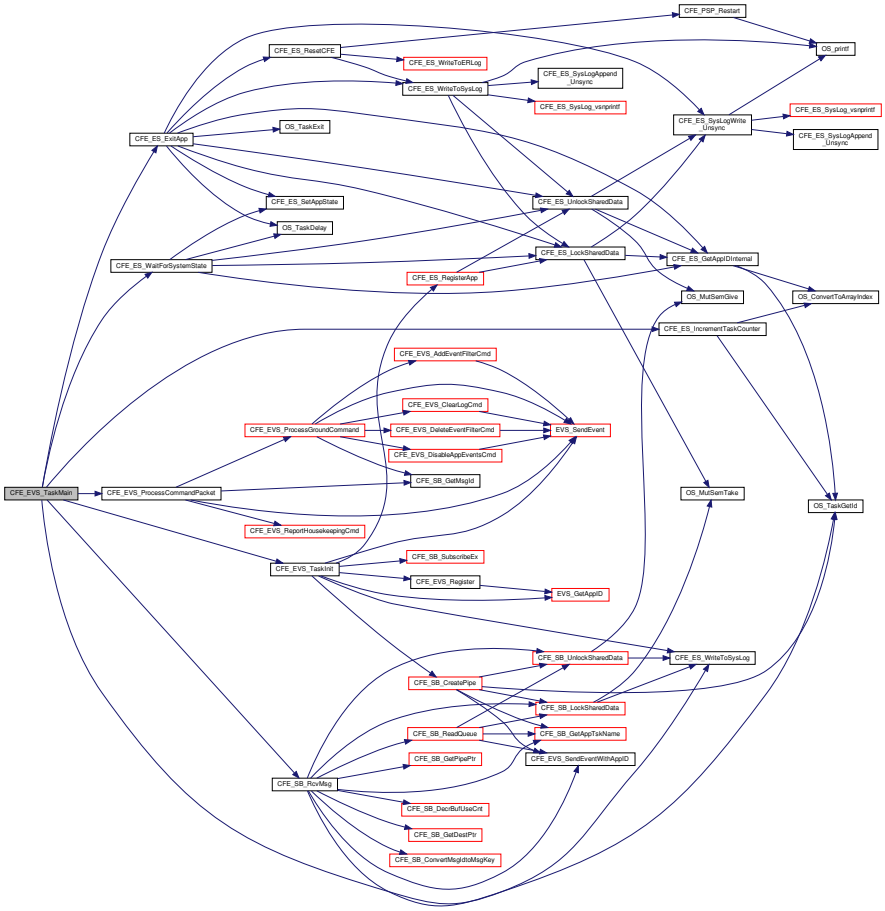
None	

Definition at line 212 of file cfe_evs_task.c.

References CFE_ES_ExitApp(), CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit,

CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, CFE_ES_SystemState_CORE_READY, CFE_ES_WaitForSystemState(), CFE_ES_WriteToSysLog(), CFE_EVS_ProcessCommandPacket(), CFE_EVS_TaskInit(), CFE_MISSION_EVS_MAIN_PERF_ID, CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_SB_PEND_FOREVER, CFE_SB_RcvMsg(), CFE_SUCCESS, and CFE_EVS_GlobalData_t::EVS_CommandPipe.

Here is the call graph for this function:



13.42.1.26 CFE_EVS_VerifyCmdLength()

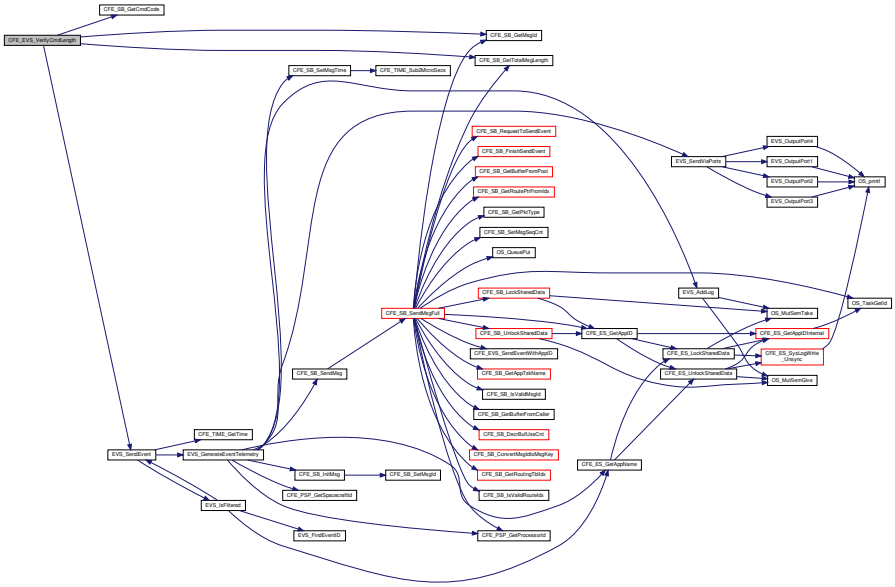
```
bool CFE_EVS_VerifyCmdLength (
            CFE_SB_MsgPtr_t Msg,
            uint16 ExpectedLength )
```

Definition at line 606 of file cfe_evs_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_LEN_ERR_EID, CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(), CFE_SB_GetTotalMsgLength(), and EVS_SendEvent().

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.1.27 CFE_EVS_WriteAppDataFileCmd()

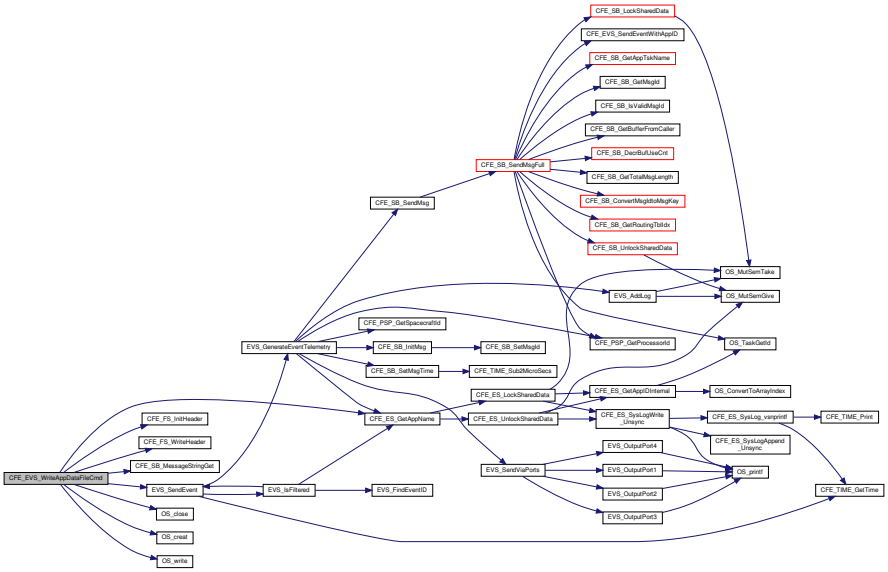
```
int32 CFE_EVS_WriteAppDataFileCmd (
            const CFE_EVS_WriteAppDataFile_t * data )
```

Definition at line 1730 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_AppDataFile_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppDataCmd_Payload_t::AppDataFilename, CFE_EVS_AppDataFile_t::AppName, EVS_AppData_t::Bin↩ Filters, CFE_ES_GetAppName(), CFE_EVS_ERR_CRDATFILE_EID, CFE_EVS_ERR_WRDATFILE_EID, CFE_EV↩ S_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_FILE_WRITE_ERROR, CFE_EVS_WRDAT_EID, CFE_FS_InitHeader(), CFE_FS_SubType_EVS_APPDATA, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_MAX↩ _APPLICATIONS, CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE, CFE_PLATFORM_EVS_MAX_EVENT_↩ FILTERS, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_AppData_t::EventCount, CFE_EVS_AppDataFile↩ _t::EventCount, EVS_AppData_t::EventTypesActiveFlag, CFE_EVS_AppDataFile_t::EventTypesActiveFlag, EVS_↩ SendEvent(), CFE_EVS_AppDataFile_t::Filters, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_write(), OS_WRITE_ONLY, CFE_EVS_WriteAppDataFile_t::Payload, and EVS_AppData↩ _t::RegisterFlag.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.42.2 Variable Documentation

13.42.2.1 CFE_EVS_GlobalData

```
CFE_EVS_GlobalData_t CFE_EVS_GlobalData
```

Definition at line 50 of file cfe_evs_task.c.

Referenced by CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetFilter(), CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), CFE_EVS_SetLogModeCmd(), CFE_EVS_↩ Unregister(), CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), EVS_ClearLog(), EVS_DisableTypes(), EVS_↩ EnableTypes(), EVS_GenerateEventTelemetry(), EVS_GetApplicationInfo(), EVS_IsFiltered(), EVS_NotRegistered(), EVS_SendEvent(), and EVS_SendViaPorts().

13.43 cfe/fsw/cfe-core/src/evs/cfe_evs_task.h File Reference

```
#include "private/cfe_private.h"
#include "private/cfe_evs_log_typedef.h"
#include "cfe_sb.h"
#include "cfe_msgids.h"
#include "cfe_es.h"
#include "osapi.h"
#include "cfe_evs_msg.h"
#include "cfe_evs_verify.h"
#include "cfe_evs.h"
#include "cfe_evs_events.h"
```

Data Structures

- struct EVS_BinFilter_t
- struct EVS_AppData_t
- struct CFE_EVS_AppDataFile_t
- struct CFE_EVS_GlobalData_t

Macros

- #define CFE_EVS_MSG_TRUNCATED '$'
- #define CFE_EVS_FREE_SLOT (-1)
- #define CFE_EVS_NO_MASK 0
- #define CFE_EVS_PIPE_DEPTH 32
- #define CFE_EVS_MSG_LIMIT 4
- #define CFE_EVS_MAX_EVENT_SEND_COUNT 65535
- #define CFE_EVS_MAX_FILTER_COUNT 65535
- #define CFE_EVS_PIPE_NAME "EVS_CMD_PIPE"
- #define CFE_EVS_UNDEF_APPID 0xFFFFFFFF
- #define CFE_EVS_MAX_PORT_MSG_LENGTH (CFE_MISSION_EVS_MAX_MESSAGE_LENGTH+OS_MA↩ X_API_NAME+30)

Functions

- int32 CFE_EVS_TaskInit (void)
- void CFE_EVS_ProcessCommandPacket (CFE_SB_MsgPtr_t EVS_MsgPtr)
- int32 CFE_EVS_ReportHousekeepingCmd (const CCSDS_CommandPacket_t *data)
- int32 CFE_EVS_NoopCmd (const CFE_EVS_Noop_t *data)
- int32 CFE_EVS_ClearLogCmd (const CFE_EVS_ClearLog_t *data)
- int32 CFE_EVS_ResetCountersCmd (const CFE_EVS_ResetCounters_t *data)
- int32 CFE_EVS_SetFilterCmd (const CFE_EVS_SetFilter_t *data)
- int32 CFE_EVS_EnablePortsCmd (const CFE_EVS_EnablePorts_t *data)
- int32 CFE_EVS_DisablePortsCmd (const CFE_EVS_DisablePorts_t *data)
- int32 CFE_EVS_EnableEventTypeCmd (const CFE_EVS_EnableEventType_t *data)
- int32 CFE_EVS_DisableEventTypeCmd (const CFE_EVS_DisableEventType_t *data)
- int32 CFE_EVS_SetEventFormatModeCmd (const CFE_EVS_SetEventFormatMode_t *data)
- int32 CFE_EVS_EnableAppEventTypeCmd (const CFE_EVS_EnableAppEventType_t *data)
- int32 CFE_EVS_DisableAppEventTypeCmd (const CFE_EVS_DisableAppEventType_t *data)
- int32 CFE_EVS_EnableAppEventsCmd (const CFE_EVS_EnableAppEvents_t *data)
- int32 CFE_EVS_DisableAppEventsCmd (const CFE_EVS_DisableAppEvents_t *data)
- int32 CFE_EVS_ResetAppCounterCmd (const CFE_EVS_ResetAppCounter_t *data)
- int32 CFE_EVS_ResetFilterCmd (const CFE_EVS_ResetFilter_t *data)
- int32 CFE_EVS_AddEventFilterCmd (const CFE_EVS_AddEventFilter_t *data)
- int32 CFE_EVS_DeleteEventFilterCmd (const CFE_EVS_DeleteEventFilter_t *data)
- int32 CFE_EVS_WriteAppDataFileCmd (const CFE_EVS_WriteAppDataFile_t *data)
- int32 CFE_EVS_ResetAllFiltersCmd (const CFE_EVS_ResetAllFilters_t *data)

Variables

- CFE_EVS_GlobalData_t CFE_EVS_GlobalData

13.43.1 Macro Definition Documentation

13.43.1.1 CFE_EVS_FREE_SLOT

```
#define CFE_EVS_FREE_SLOT (-1)
```

Definition at line 59 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), and CFE_EVS_Register().

13.43.1.2 CFE_EVS_MAX_EVENT_SEND_COUNT

```
#define CFE_EVS_MAX_EVENT_SEND_COUNT 65535
```

Definition at line 63 of file cfe_evs_task.h.

Referenced by EVS_GenerateEventTelemetry().

13.43.1.3 CFE_EVS_MAX_FILTER_COUNT

```
#define CFE_EVS_MAX_FILTER_COUNT 65535
```

Definition at line 64 of file cfe_evs_task.h.

Referenced by EVS_IsFiltered().

13.43.1.4 CFE_EVS_MAX_PORT_MSG_LENGTH

```
#define CFE_EVS_MAX_PORT_MSG_LENGTH (CFE_MISSION_EVS_MAX_MESSAGE_LENGTH+OS_MAX_API_NAME+30)
```

Definition at line 67 of file cfe_evs_task.h.

Referenced by EVS_SendViaPorts().

13.43.1.5 CFE_EVS_MSG_LIMIT

```
#define CFE_EVS_MSG_LIMIT 4
```

Definition at line 62 of file cfe_evs_task.h.

Referenced by CFE_EVS_TaskInit().

13.43.1.6 CFE_EVS_MSG_TRUNCATED

```
#define CFE_EVS_MSG_TRUNCATED '$'
```

Definition at line 58 of file cfe_evs_task.h.

Referenced by EVS_GenerateEventTelemetry().

13.43.1.7 CFE_EVS_NO_MASK

```
#define CFE_EVS_NO_MASK 0
```

Definition at line 60 of file cfe_evs_task.h.

Referenced by CFE_EVS_DeleteEventFilterCmd().

13.43.1.8 CFE_EVS_PIPE_DEPTH

```
#define CFE_EVS_PIPE_DEPTH 32
```

Definition at line 61 of file cfe_evs_task.h.

Referenced by CFE_EVS_TaskInit().

13.43.1.9 CFE_EVS_PIPE_NAME

```
#define CFE_EVS_PIPE_NAME "EVS_CMD_PIPE"
```

Definition at line 65 of file cfe_evs_task.h.

Referenced by CFE_EVS_TaskInit().

13.43.1.10 CFE_EVS_UNDEF_APPID

```
#define CFE_EVS_UNDEF_APPID 0xFFFFFFFF
```

Definition at line 66 of file cfe_evs_task.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEvents↩Cmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EarlyInit(), CFE_EVS_EnableAppEventsCmd(), CFE_E↩VS_EnableAppEventTypeCmd(), CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SendEvent(), CFE_EVS_SendTimedEvent(), CFE_EVS_SetFilterCmd(), and CFE_EVS_Unregister().

13.43.2 Function Documentation

13.43.2.1 CFE_EVS_AddEventFilterCmd()

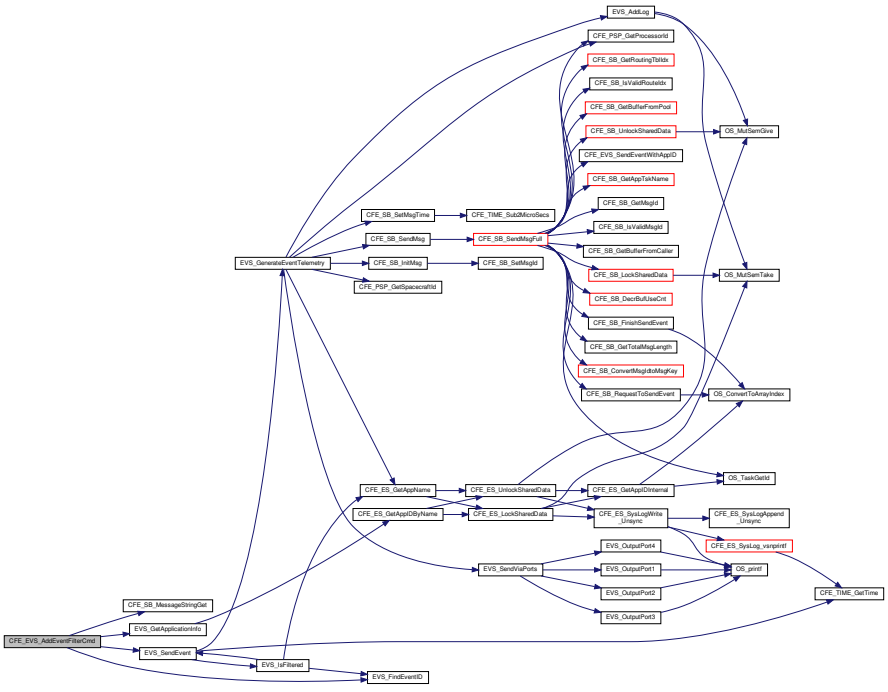
```
int32 CFE_EVS_AddEventFilterCmd (
            const CFE_EVS_AddEventFilter_t * data )
```

Definition at line 1555 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::AppName, EVS_AppData_t::BinFilters, CFE_EVS_ADD_EVENT_FILTER_CC, CFE_EVS_ADDFILTER_EID, CFE_EVS_APP_FILTER_OVERLOAD, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_MAXREGSFILTER_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_FILTERED_EID, CFE_EVS_EVT_NOT_REGISTERED, CFE_EVS_FREE_SLOT, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_BinFilter_t::Count, EVS_BinFilter_t::EventID, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::EventID, EVS_FindEventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), EVS_BinFilter_t::Mask, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::Mask, NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameEventIDMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.2 CFE_EVS_ClearLogCmd()

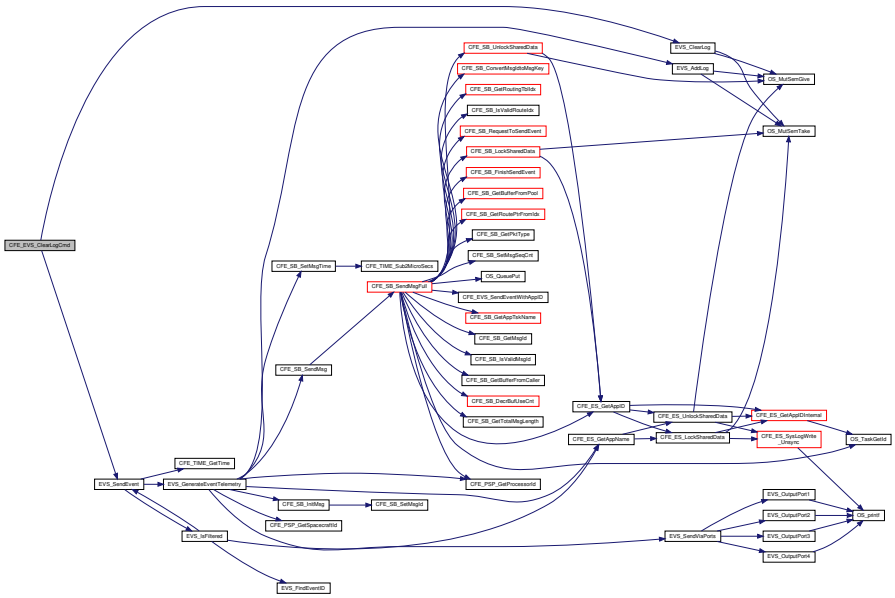
```
int32 CFE_EVS_ClearLogCmd (
            const CFE_EVS_ClearLog_t * data )
```

Definition at line 656 of file cfe_evs_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_FUNCTION_DISABLED, CFE_EVS_NO_LOGCLR_EID, CFE_SUCCESS, EVS_ClearLog(), EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.3 CFE_EVS_DeleteEventFilterCmd()

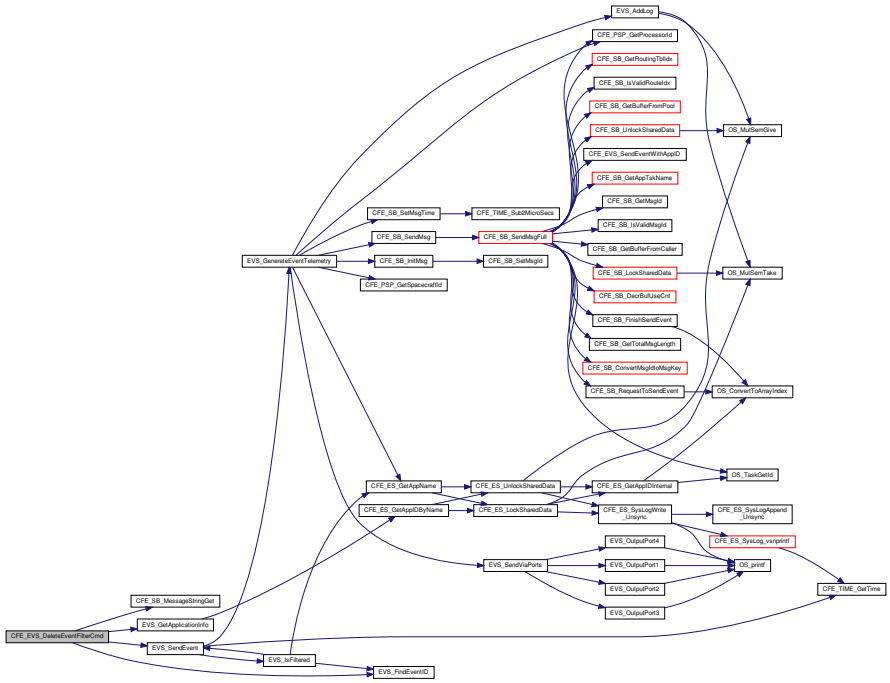
```
int32 CFE_EVS_DeleteEventFilterCmd (
            const CFE_EVS_DeleteEventFilter_t * data )
```

Definition at line 1650 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDCmd_Payload_t::AppName, EVS_AppData_t::BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_DELETE_EVENT_FILTER_CC, CFE_EVS_DELFILTER_EID, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_EVTIDNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_NOT_REGISTERED, CFE_EVS_FREE_SLOT, CFE_EVS_NO_MASK, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_BinFilter_t::Count, EVS_BinFilter_t::EventID, CFE_EVS_AppNameEventIDCmd_Payload_t::EventID, EVS_FindEventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), EVS_BinFilter_t::Mask, NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameEventIDCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.4 CFE_EVS_DisableAppEventsCmd()

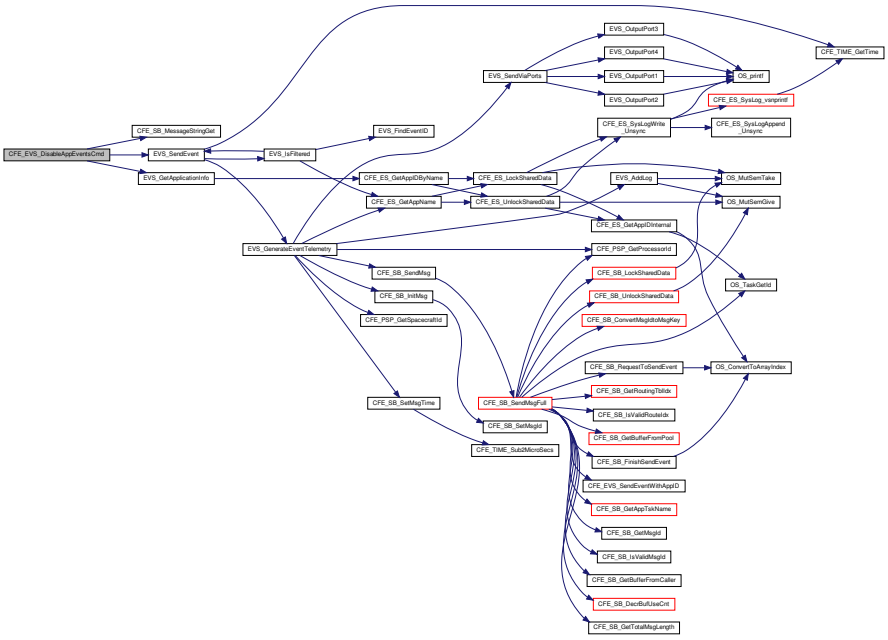
```
int32 CFE_EVS_DisableAppEventsCmd (
            const CFE_EVS_DisableAppEvents_t * data )
```

Definition at line 1299 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_DISAPPEVT_EID, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.5 CFE_EVS_DisableAppEventTypeCmd()

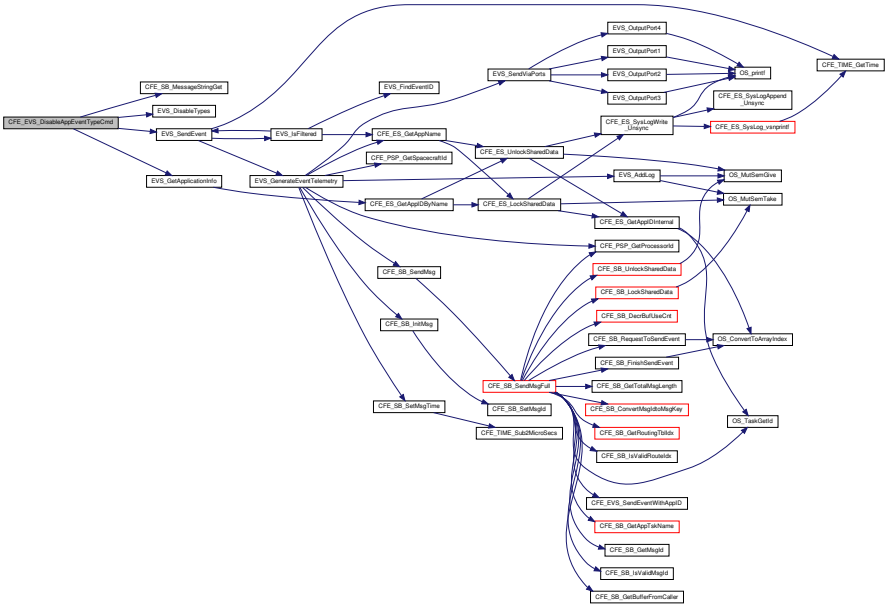
```
int32 CFE_EVS_DisableAppEventTypeCmd (
            const CFE_EVS_DisableAppEventType_t * data )
```

Definition at line 1168 of file cfe_evs_task.c.

References CFE_EVS_AppNameBitMaskCmd_Payload_t::AppName, CFE_EVS_AppNameBitMaskCmd_Payload_t::BitMask, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISAPPENTTYPE_EID, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_DisableTypes(), EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameBitMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.6 CFE_EVS_DisableEventTypeCmd()

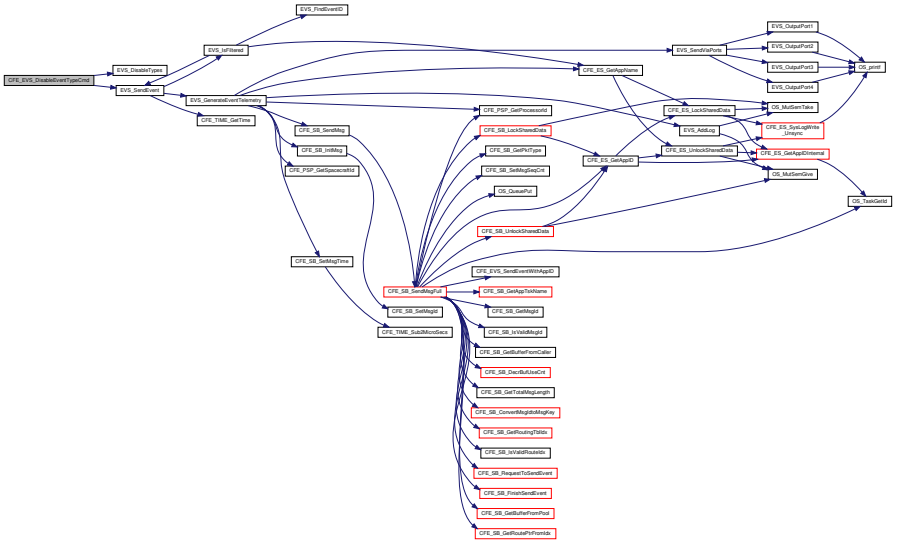
```
int32 CFE_EVS_DisableEventTypeCmd (
            const CFE_EVS_DisableEventType_t * data )
```

Definition at line 1007 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_DISABLE↩_EVENT_TYPE_CC, CFE_EVS_DISEVTTYPE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_Event↩Type_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_PLATFORM_ES_MAX_A↩PPLICATIONS, CFE_SUCCESS, EVS_DisableTypes(), EVS_SendEvent(), CFE_EVS_BitMaskCmd_t::Payload, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.7 CFE_EVS_DisablePortsCmd()

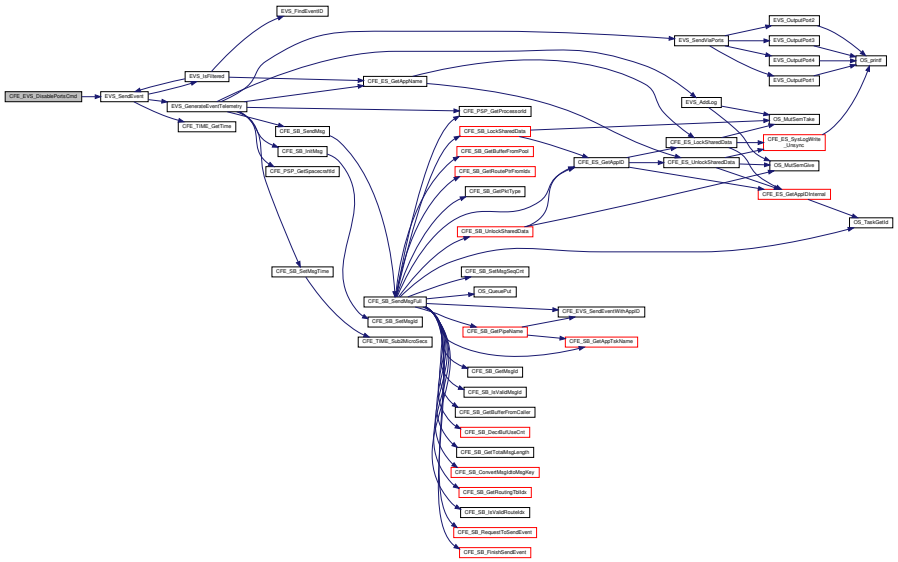
```
int32 CFE_EVS_DisablePortsCmd (
            const CFE_EVS_DisablePorts_t * data )
```

Definition at line 901 of file cfe_evs_task.c.

References CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_DISABLE_PORTS_CC, CFE_EVS_DISPOR↩T_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT, C↩FE_EVS_PORT4_BIT, CFE_SUCCESS, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_↩HousekeepingTlm_Payload_t::OutputPort, CFE_EVS_BitMaskCmd_t::Payload, and CFE_EVS_HousekeepingTlm_t↩::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.8 CFE_EVS_EnableAppEventsCmd()

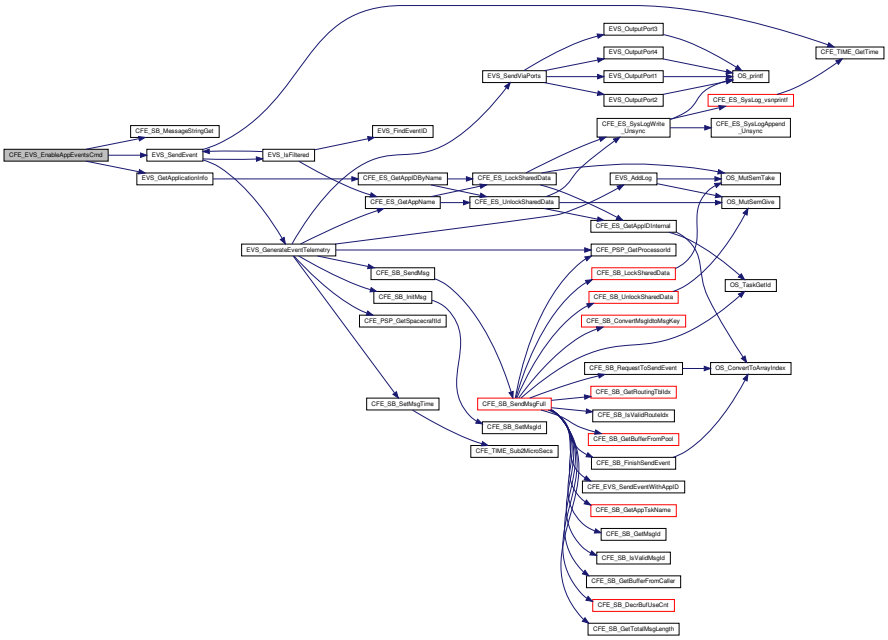
```
int32 CFE_EVS_EnableAppEventsCmd (
            const CFE_EVS_EnableAppEvents_t * data )
```

Definition at line 1241 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ENAAPPEVT_EID, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.9 CFE_EVS_EnableAppEventTypeCmd()

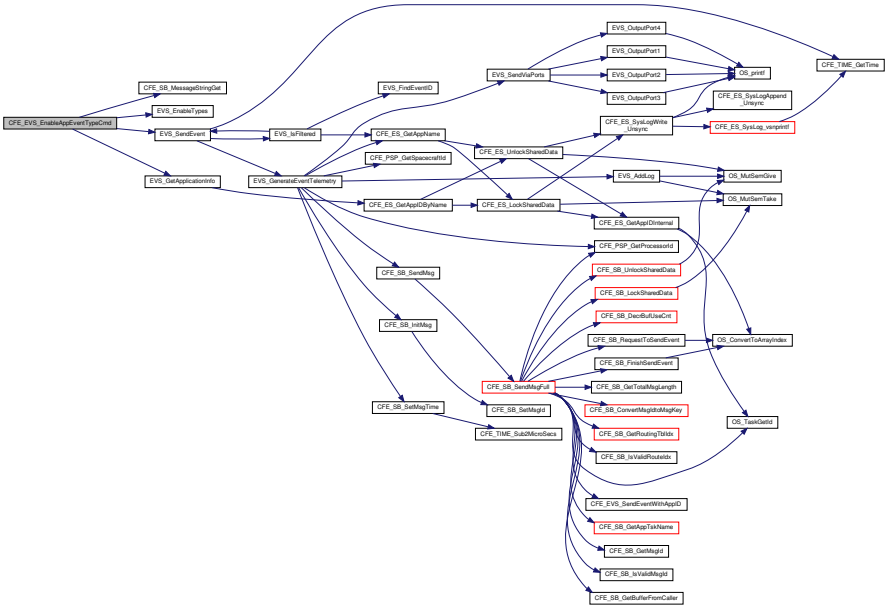
```
int32 CFE_EVS_EnableAppEventTypeCmd (
            const CFE_EVS_EnableAppEventType_t * data )
```

Definition at line 1094 of file cfe_evs_task.c.

References CFE_EVS_AppNameBitMaskCmd_Payload_t::AppName, CFE_EVS_AppNameBitMaskCmd_Payload_t::BitMask, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ENAAPPEVTTYPE_EID, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_UNDEF_APPID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_EnableTypes(), EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameBitMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.10 CFE_EVS_EnableEventTypeCmd()

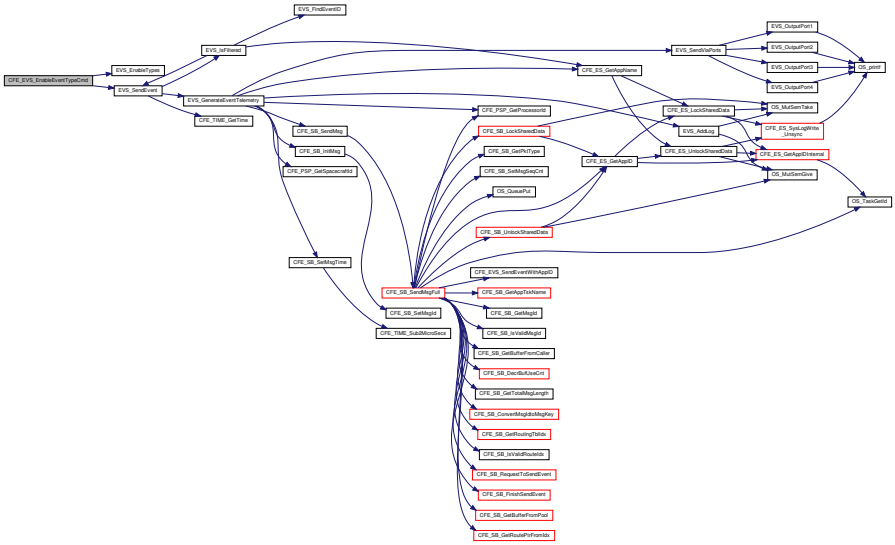
```
int32 CFE_EVS_EnableEventTypeCmd (
            const CFE_EVS_EnableEventType_t * data )
```

Definition at line 959 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_ENABLE↩
_EVENT_TYPE_CC, CFE_EVS_ENAEVTTYPE_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_Event↩
Type_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_INVALID_PARAMETER, CFE_PLATFORM_ES_MAX_↩
APPLICATIONS, CFE_SUCCESS, EVS_EnableTypes(), EVS_SendEvent(), CFE_EVS_BitMaskCmd_t::Payload, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.11 CFE_EVS_EnablePortsCmd()

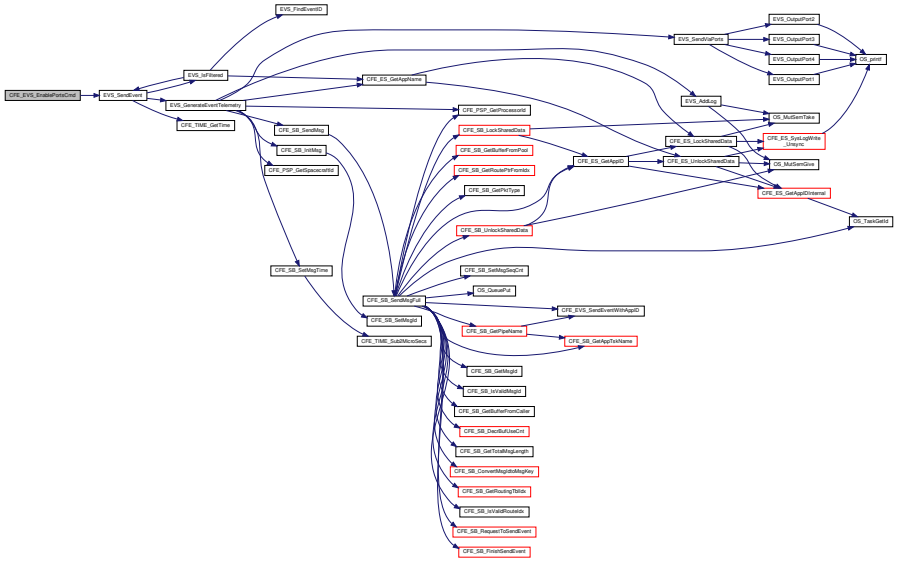
```
int32 CFE_EVS_EnablePortsCmd (
            const CFE_EVS_EnablePorts_t * data )
```

Definition at line 845 of file cfe_evs_task.c.

References CFE_EVS_BitMaskCmd_Payload_t::BitMask, CFE_EVS_ENABLE_PORTS_CC, CFE_EVS_ENAPO↩RT_EID, CFE_EVS_ERR_INVALID_BITMASK_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERR↩OR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT, CFE_EVS_PORT4_BIT, CFE_SUCCESS, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_↩HousekeepingTlm_Payload_t::OutputPort, CFE_EVS_BitMaskCmd_t::Payload, and CFE_EVS_HousekeepingTlm_t↩::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.12 CFE_EVS_NoopCmd()

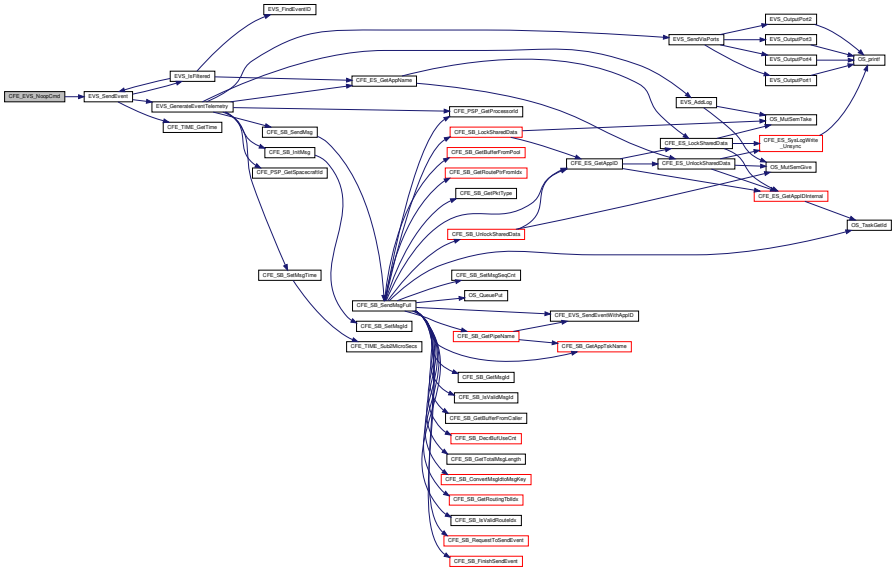
```
int32 CFE_EVS_NoopCmd (
            const CFE_EVS_Noop_t * data )
```

Definition at line 639 of file cfe_evs_task.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_NOOP_EID, CFE_MAJOR_VERSION, CFE_MINOR↩
_VERSION, CFE_MISSION_REV, CFE_REVISION, CFE_SUCCESS, and EVS_SendEvent().

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.13 CFE_EVS_ProcessCommandPacket()

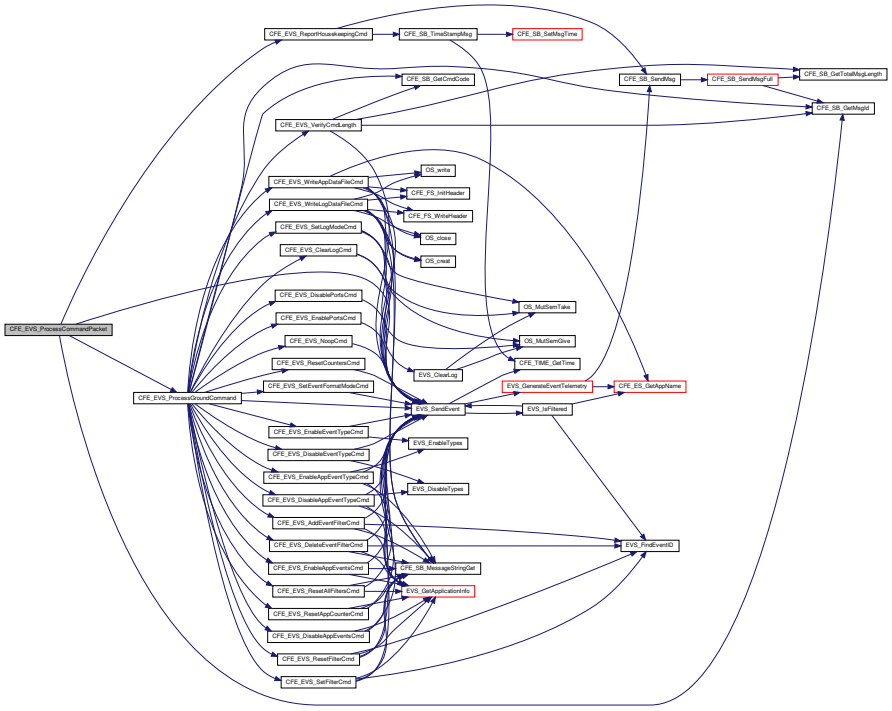
```
void CFE_EVS_ProcessCommandPacket (
            CFE_SB_MsgPtr_t EVS_MsgPtr )
```

Definition at line 355 of file cfe_evs_task.c.

References CFE_EVS_CMD_MID, CFE_EVS_ERR_MSGID_EID, CFE_EVS_EventType_ERROR, CFE_EVS_↩ ProcessGroundCommand(), CFE_EVS_ReportHousekeepingCmd(), CFE_EVS_SEND_HK_MID, CFE_SB_Get↩ MsgId(), CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter, EVS_SendEvent(), CFE_EVS_Global↩ Data_t::EVS_TlmPkt, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_TaskMain().

Here is the call graph for this function:



13.43.2.14 CFE_EVS_ReportHousekeepingCmd()

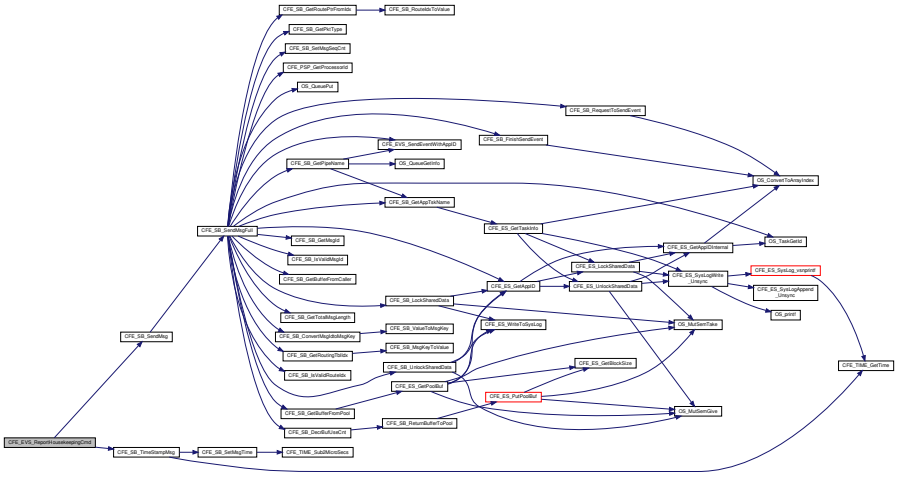
```
int32 CFE_EVS_ReportHousekeepingCmd (
            const CCSDS_CommandPacket_t * data )
```

Definition at line 684 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_HousekeepingTlm_Payload_t::AppData, CFE_EVS_AppTlmData_t::AppEnableStatus, CFE_EVS_AppTlmData_t::AppID, CFE_EVS_AppTlmData_t::AppMessageSentCounter, CFE_MISSION_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_STATUS_NO_COUNTER_INCREMENT, EVS_AppData_t::EventCount, CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_HousekeepingTlm_Payload_t::LogFullFlag, CFE_EVS_Log_t::LogMode, CFE_EVS_HousekeepingTlm_Payload_t::LogMode, CFE_EVS_Log_t::LogOverflowCounter, CFE_EVS_HousekeepingTlm_Payload_t::LogOverflowCounter, CFE_EVS_HousekeepingTlm_Payload_t::Payload, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_ProcessCommandPacket().

Here is the call graph for this function:



13.43.2.15 CFE_EVS_ResetAllFiltersCmd()

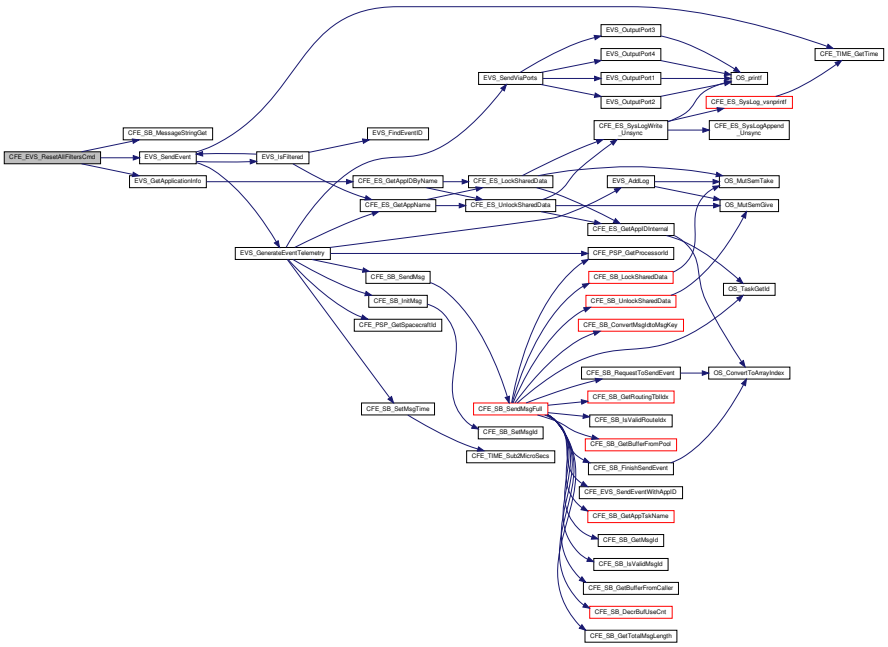
```
int32 CFE_EVS_ResetAllFiltersCmd (
            const CFE_EVS_ResetAllFilters_t * data )
```

Definition at line 1493 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, EVS_AppData_t::↩
BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_APPNORE↩
GS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType↩
_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_RSTALLFILTER_EID, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SB_MessageStringGet(), CFE↩
_SUCCESS, EVS_BinFilter_t::Count, EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.16 CFE_EVS_ResetAppCounterCmd()

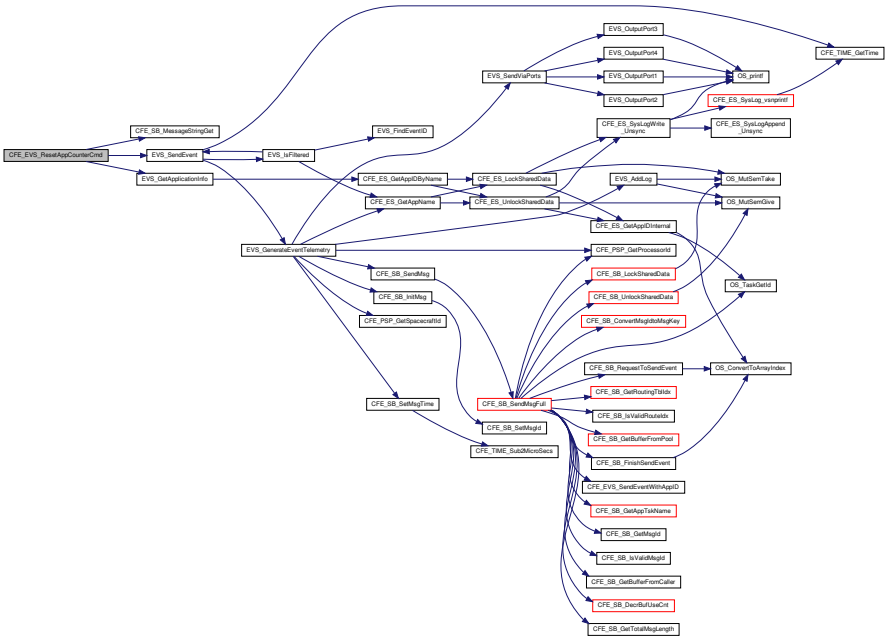
```
int32 CFE_EVS_ResetAppCounterCmd (
            const CFE_EVS_ResetAppCounter_t * data )
```

Definition at line 1358 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameCmd_Payload_t::AppName, CFE_EVS_APP_↩ ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_APPNOREGS_EID, CFE_EVS_ERR_I↩ LLAPPIDRANGE_EID, CFE_EVS_ERR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_Event↩ Type_ERROR, CFE_EVS_RESET_APP_COUNTER_CC, CFE_EVS_RSTEVTCNT_EID, CFE_EVS_UNDEF_APP↩ ID, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_AppData_t::EventCount, EVS_GetApplicationInfo(), EVS↩ _SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.17 CFE_EVS_ResetCountersCmd()

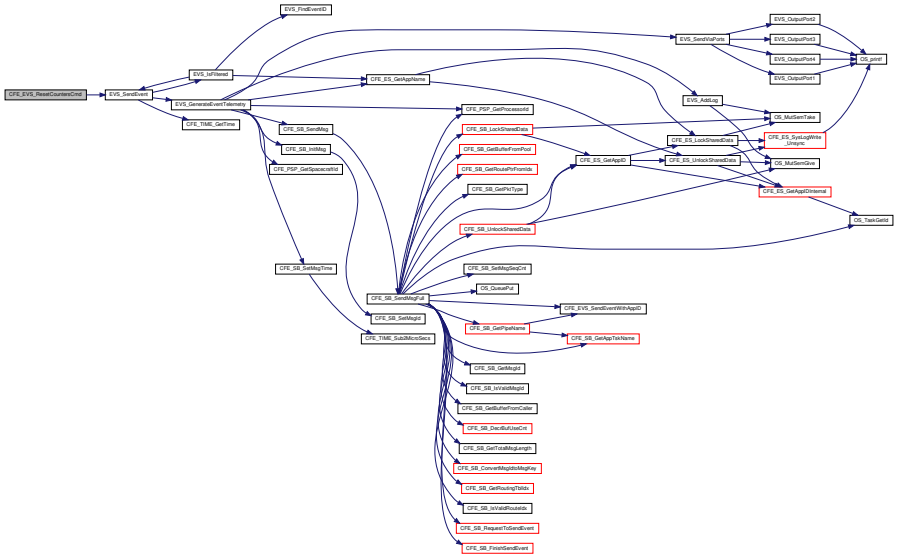
```
int32 CFE_EVS_ResetCountersCmd (
            const CFE_EVS_ResetCounters_t * data )
```

Definition at line 736 of file cfe_evs_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_RSTCNT_EID, CFE_STATUS_NO_COUNTER_INCREMENT, CFE_EVS_HousekeepingTlm_Payload_t::CommandCounter, CFE_EVS_HousekeepingTlm_Payload_t::CommandErrorCounter, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_HousekeepingTlm_Payload_t::MessageSendCounter, CFE_EVS_HousekeepingTlm_Payload_t::MessageTruncCounter, CFE_EVS_HousekeepingTlm_Payload_t::Payload, and CFE_EVS_HousekeepingTlm_Payload_t::UnregisteredAppCounter.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.18 CFE_EVS_ResetFilterCmd()

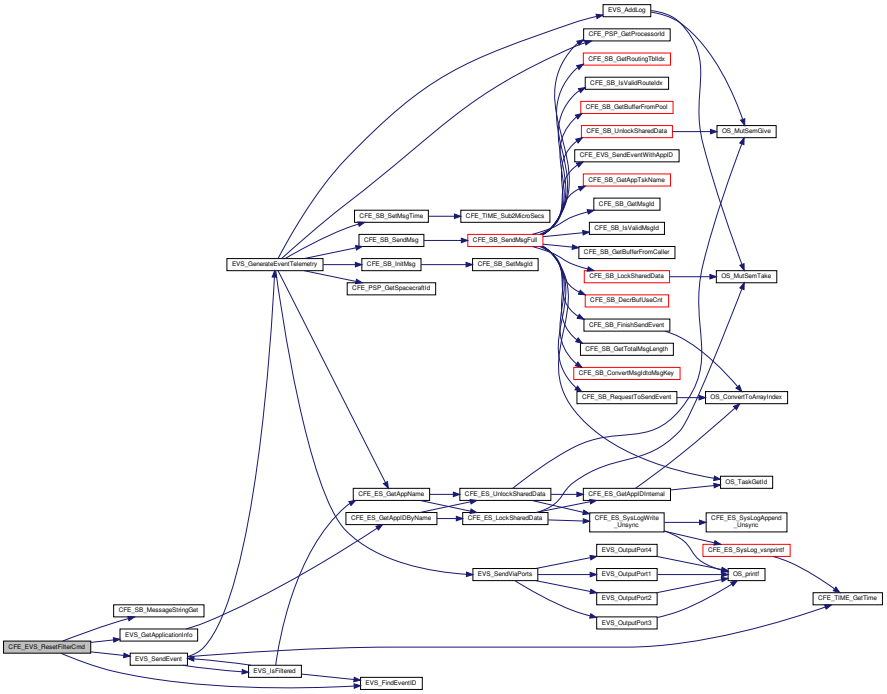
```
int32 CFE_EVS_ResetFilterCmd (
            const CFE_EVS_ResetFilter_t * data )
```

Definition at line 1417 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDCmd_Payload_t::AppName, EVS_App↩
Data_t::BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR_AP↩
PNOREGS_EID, CFE_EVS_ERR_EVTIDNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_ERR↩
_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_NOT_R↩
EGISTERED, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RSTFILTER_EID, CFE_EVS_UNDEF_APPID, CFE_SB_↩
MessageStringGet(), CFE_SUCCESS, EVS_BinFilter_t::Count, CFE_EVS_AppNameEventIDCmd_Payload_t::Event↩
ID, EVS_FindEventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), NULL, OS_MAX_API_NAME, and CFE_EVS↩
_AppNameEventIDCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.19 CFE_EVS_SetEventFormatModeCmd()

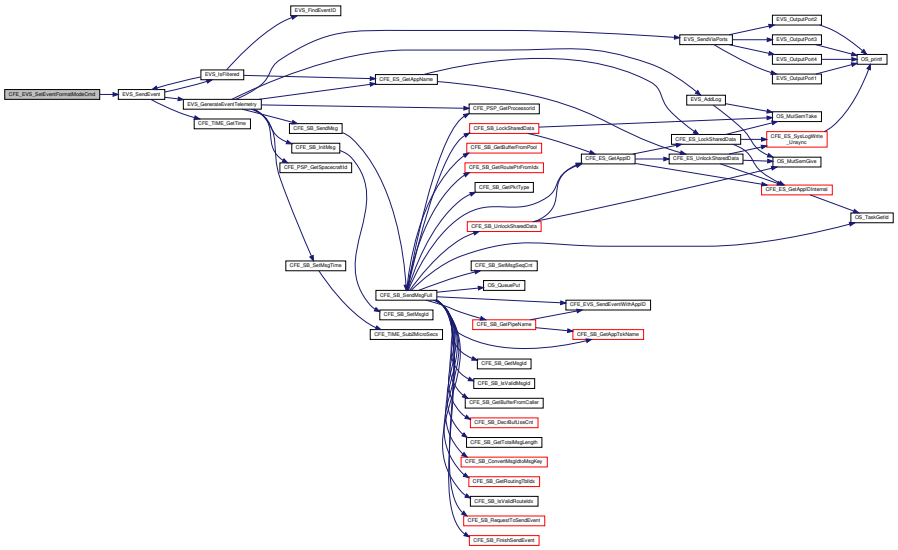
```
int32 CFE_EVS_SetEventFormatModeCmd (
            const CFE_EVS_SetEventFormatMode_t * data )
```

Definition at line 1056 of file cfe_evs_task.c.

References CFE_EVS_ERR_ILLEGALFMTMOD_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERR↩OR, CFE_EVS_INVALID_PARAMETER, CFE_EVS_MsgFormat_LONG, CFE_EVS_MsgFormat_SHORT, CFE_EV↩S_SETEVTFMTMOD_EID, CFE_SUCCESS, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS↩_HousekeepingTlm_Payload_t::MessageFormatMode, CFE_EVS_SetEventFormatMode_Payload_t::MsgFormat, C↩FE_EVS_SetEventFormatMode_t::Payload, and CFE_EVS_HousekeepingTlm_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.20 CFE_EVS_SetFilterCmd()

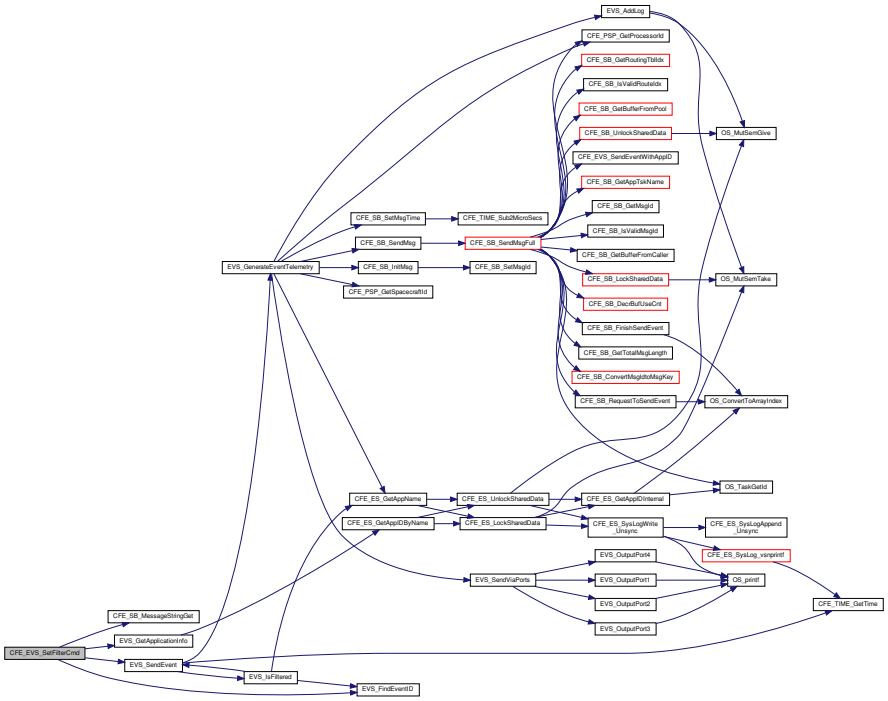
```
int32 CFE_EVS_SetFilterCmd (
            const CFE_EVS_SetFilter_t * data )
```

Definition at line 766 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::AppName, EVS↩
_AppData_t::BinFilters, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_ERR↩
_APPNOREGS_EID, CFE_EVS_ERR_EVTIDNOREGS_EID, CFE_EVS_ERR_ILLAPPIDRANGE_EID, CFE_EVS_E↩
RR_NOAPPIDFOUND_EID, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EVT_NOT↩
_REGISTERED, CFE_EVS_SET_FILTER_CC, CFE_EVS_SETFILTERMSK_EID, CFE_EVS_UNDEF_APPID, CFE↩
_SB_MessageStringGet(), CFE_SUCCESS, CFE_EVS_AppNameEventIDMaskCmd_Payload_t::EventID, EVS_Find↩
EventID(), EVS_GetApplicationInfo(), EVS_SendEvent(), EVS_BinFilter_t::Mask, CFE_EVS_AppNameEventIDMask↩
Cmd_Payload_t::Mask, NULL, OS_MAX_API_NAME, and CFE_EVS_AppNameEventIDMaskCmd_t::Payload.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.2.21 CFE_EVS_TaskInit()

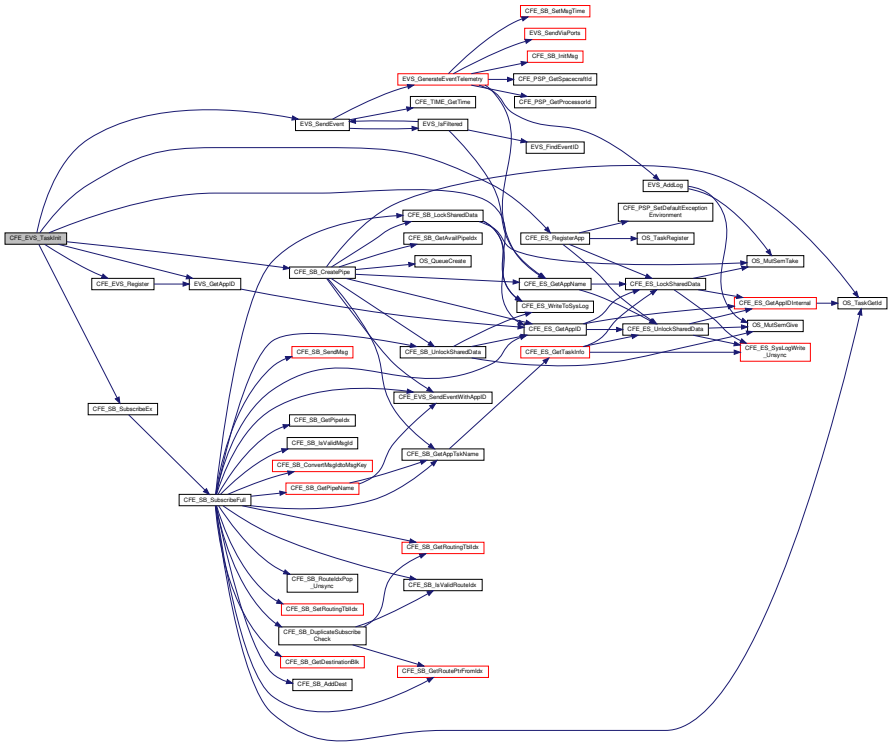
```
int32 CFE_EVS_TaskInit (
            void  )
```

Definition at line 279 of file cfe_evs_task.c.

References CFE_ES_RegisterApp(), CFE_ES_WriteToSysLog(), CFE_EVS_CMD_MID, CFE_EVS_EventFilter_B↩INARY, CFE_EVS_EventType_INFORMATION, CFE_EVS_MSG_LIMIT, CFE_EVS_PIPE_DEPTH, CFE_EVS_PI↩PE_NAME, CFE_EVS_Register(), CFE_EVS_SEND_HK_MID, CFE_EVS_STARTUP_EID, CFE_MAJOR_VERSI↩ON, CFE_MINOR_VERSION, CFE_MISSION_REV, CFE_REVISION, CFE_SB_CreatePipe(), CFE_SB_Default_Qos, CFE_SB_SubscribeEx(), CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_AppID, CFE_EVS_GlobalData_t::EVS_↩CommandPipe, EVS_GetAppID(), EVS_SendEvent(), and NULL.

Referenced by CFE_EVS_TaskMain().

Here is the call graph for this function:



13.43.2.22 CFE_EVS_WriteAppDataFileCmd()

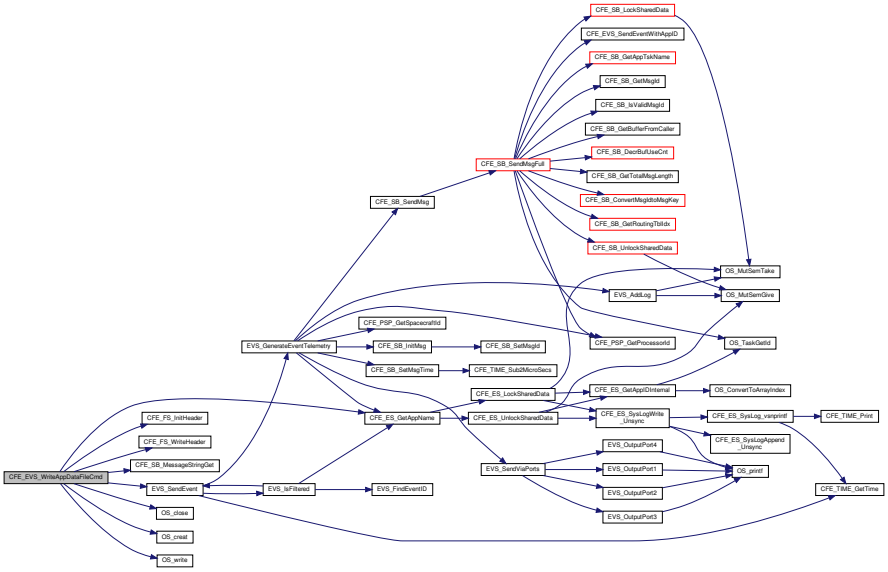
```
int32 CFE_EVS_WriteAppDataFileCmd (
            const CFE_EVS_WriteAppDataFile_t * data )
```

Definition at line 1730 of file cfe_evs_task.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_AppDataFile_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, CFE_EVS_AppDataCmd_Payload_t::AppDataFilename, CFE_EVS_AppDataFile_t::AppName, EVS_AppData_t::Bin↩ Filters, CFE_ES_GetAppName(), CFE_EVS_ERR_CRDATFILE_EID, CFE_EVS_ERR_WRDATFILE_EID, CFE_EV↩ S_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_FILE_WRITE_ERROR, CFE_EVS_WRDAT_EID, CFE_FS_InitHeader(), CFE_FS_SubType_EVS_APPDATA, CFE_FS_WriteHeader(), CFE_PLATFORM_ES_MAX↩ _APPLICATIONS, CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE, CFE_PLATFORM_EVS_MAX_EVENT_↩ FILTERS, CFE_SB_MessageStringGet(), CFE_SUCCESS, EVS_AppData_t::EventCount, CFE_EVS_AppDataFile↩ _t::EventCount, EVS_AppData_t::EventTypesActiveFlag, CFE_EVS_AppDataFile_t::EventTypesActiveFlag, EVS_↩ SendEvent(), CFE_EVS_AppDataFile_t::Filters, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_MAX_API_NAME, OS_MAX_PATH_LEN, OS_write(), OS_WRITE_ONLY, CFE_EVS_WriteAppDataFile_t::Payload, and EVS_AppData↩ _t::RegisterFlag.

Referenced by CFE_EVS_ProcessGroundCommand().

Here is the call graph for this function:



13.43.3 Variable Documentation

13.43.3.1 CFE_EVS_GlobalData

```
CFE_EVS_GlobalData_t CFE_EVS_GlobalData
```

Definition at line 50 of file cfe_evs_task.c.

Referenced by CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetFilter(), CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), CFE_EVS_SetLogModeCmd(), CFE_EVS_↩ Unregister(), CFE_EVS_WriteLogDataFileCmd(), EVS_AddLog(), EVS_ClearLog(), EVS_DisableTypes(), EVS_↩ EnableTypes(), EVS_GenerateEventTelemetry(), EVS_GetApplicationInfo(), EVS_IsFiltered(), EVS_NotRegistered(), EVS_SendEvent(), and EVS_SendViaPorts().

13.44 cfe/fsw/cfe-core/src/evs/cfe_evs_utils.c File Reference

```
#include "cfe_evs.h"
#include "cfe_evs_log.h"
#include "cfe_evs_task.h"
#include "cfe_evs_utils.h"
#include <stdio.h>
#include <string.h>
#include "cfe_error.h"
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
#include "cfe_sb.h"
#include "cfe_es.h"
```

Functions

- void EVS_SendViaPorts (CFE_EVS_LongEventTlm_t *EVS_PktPtr)
- void EVS_OutputPort1 (char *Message)
- void EVS_OutputPort2 (char *Message)
- void EVS_OutputPort3 (char *Message)
- void EVS_OutputPort4 (char *Message)
- int32 EVS_GetAppID (uint32 *AppIdPtr)
- int32 EVS_GetApplicationInfo (uint32 *pAppID, const char *pAppName)
- int32 EVS_NotRegistered (uint32 AppID)
- bool EVS_IsFiltered (uint32 AppID, uint16 EventID, uint16 EventType)
- EVS_BinFilter_t * EVS_FindEventID (int16 EventID, EVS_BinFilter_t *FilterArray)
- void EVS_EnableTypes (uint8 BitMask, uint32 AppID)
- void EVS_DisableTypes (uint8 BitMask, uint32 AppID)
- void EVS_GenerateEventTelemetry (uint32 AppID, uint16 EventID, uint16 EventType, const CFE_TIME_SysTime_t *TimeStamp, const char *MsgSpec, va_list ArgPtr)
- int32 EVS_SendEvent (uint16 EventID, uint16 EventType, const char *Spec,...)

13.44.1 Function Documentation

13.44.1.1 EVS_DisableTypes()

```
void EVS_DisableTypes (
            uint8 BitMask,
            uint32 AppID )
```

Definition at line 336 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_CRITICAL_BIT, CFE_EVS_DEBUG_BIT, CFE_EVS_ERROR_BIT, CFE_EVS_GlobalData, CFE_EVS_INFORMATION_BIT, and EVS_AppData_t::EventTypesActiveFlag.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), and CFE_EVS_DisableEventTypeCmd().

13.44.1.2 EVS_EnableTypes()

```
void EVS_EnableTypes (
            uint8 BitMask,
            uint32 AppID )
```

Definition at line 316 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_CRITICAL_BIT, CFE_EVS_DEBUG_BIT, CFE_EVS_ERROR_BIT, CFE_EVS_GlobalData, CFE_EVS_INFORMATION_BIT, and EVS_AppData_t::EventTypesActiveFlag.

Referenced by CFE_EVS_EnableAppEventTypeCmd(), and CFE_EVS_EnableEventTypeCmd().

13.44.1.3 EVS_FindEventID()

```
EVS_BinFilter_t* EVS_FindEventID (
            int16 EventID,
            EVS_BinFilter_t * FilterArray )
```

Definition at line 289 of file cfe_evs_utils.c.

References CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, and NULL.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_ResetFilter(), CFE_↩EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), and EVS_IsFiltered().

13.44.1.4 EVS_GenerateEventTelemetry()

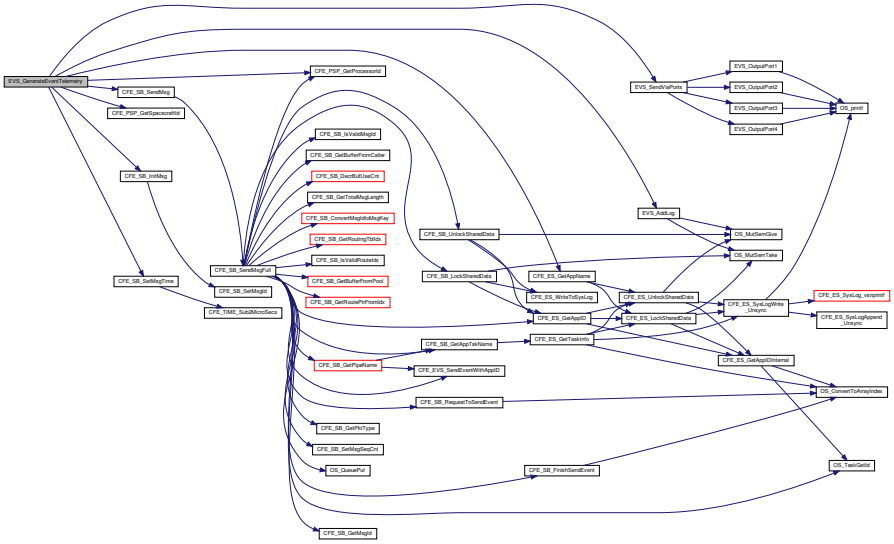
```
void EVS_GenerateEventTelemetry (
            uint32 AppID,
            uint16 EventID,
            uint16 EventType,
            const CFE_TIME_SysTime_t * TimeStamp,
            const char * MsgSpec,
            va_list ArgPtr )
```

Definition at line 360 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_PacketID_t::AppName, CFE_ES_GetAppName(), CFE↩_EVS_GlobalData, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_MAX_EVENT_SEND_COUNT, CFE_EVS_↩MSG_TRUNCATED, CFE_EVS_MsgFormat_LONG, CFE_EVS_MsgFormat_SHORT, CFE_EVS_SHORT_EVENT↩_MSG_MID, CFE_PSP_GetProcessorId(), CFE_PSP_GetSpacecraftId(), CFE_SB_InitMsg(), CFE_SB_SendMsg(), CFE_SB_SetMsgTime(), EVS_AppData_t::EventCount, CFE_EVS_PacketID_t::EventID, CFE_EVS_PacketID_t::↩EventType, EVS_AddLog(), EVS_SendViaPorts(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_LongEventTlm↩_Payload_t::Message, CFE_EVS_HousekeepingTlm_Payload_t::MessageFormatMode, CFE_EVS_Housekeeping↩Tlm_Payload_t::MessageSendCounter, CFE_EVS_HousekeepingTlm_Payload_t::MessageTruncCounter, CFE_EVS↩_LongEventTlm_Payload_t::PacketID, CFE_EVS_HousekeepingTlm_t::Payload, CFE_EVS_LongEventTlm_t::Payload, CFE_EVS_PacketID_t::ProcessorID, and CFE_EVS_PacketID_t::SpacecraftID.

Referenced by CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), and EV↩S_SendEvent().

Here is the call graph for this function:



13.44.1.5 EVS_GetAppID()

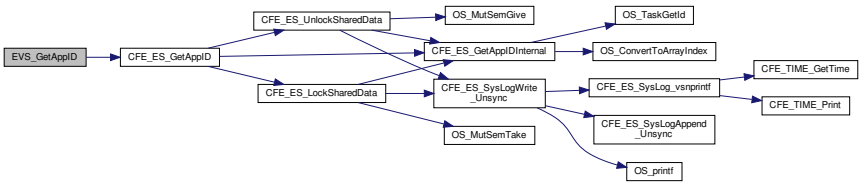
```
int32 EVS_GetAppID (
            uint32 * AppIdPtr )
```

Definition at line 67 of file cfe_evs_utils.c.

References CFE_ES_GetAppID(), CFE_EVS_APP_ILLEGAL_APP_ID, CFE_PLATFORM_ES_MAX_APPLICATIONS, and CFE_SUCCESS.

Referenced by CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetFilter(), CFE_EVS_SendEvent(), CFE_EVS_SendTimedEvent(), CFE_EVS_TaskInit(), and CFE_EVS_Unregister().

Here is the call graph for this function:



13.44.1.6 EVS_GetApplicationInfo()

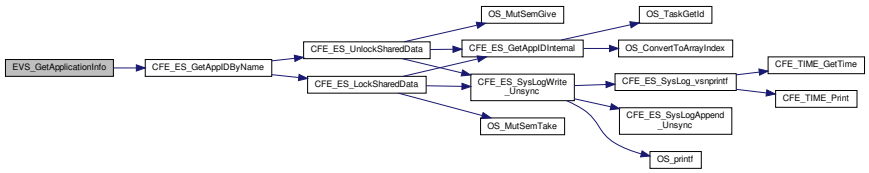
```
int32 EVS_GetApplicationInfo (
            uint32 * pAppID,
            const char * pAppName )
```

Definition at line 99 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_ES_ERR_BUFFER, CFE_ES_GetAppIDByName(), CFE_EVS_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_GlobalData, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, NULL, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

Here is the call graph for this function:



13.44.1.7 EVS_IsFiltered()

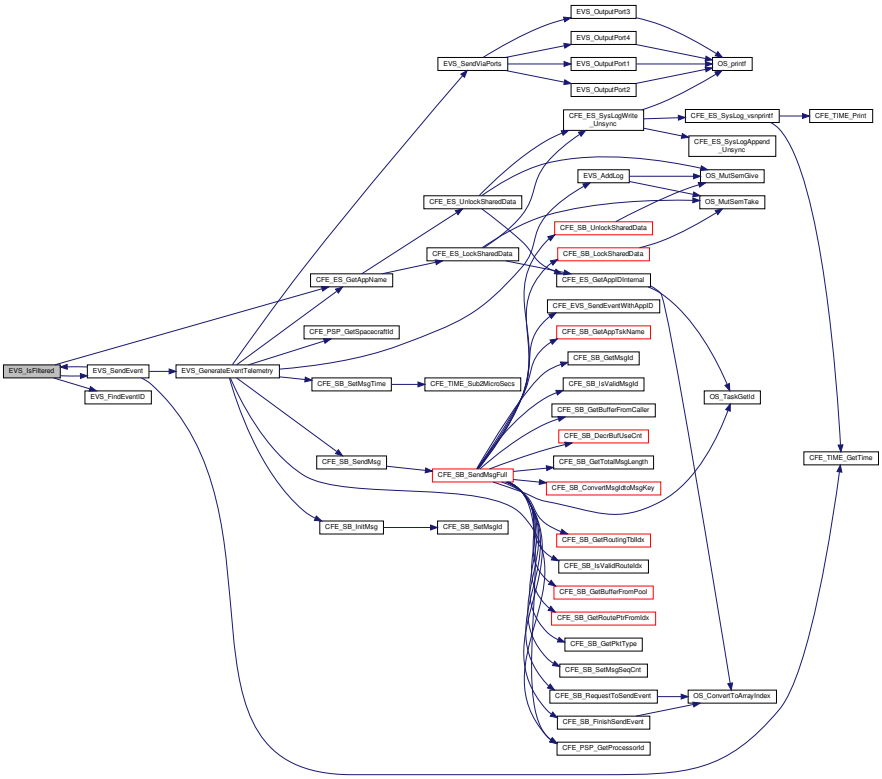
```
bool EVS_IsFiltered (
            uint32 AppID,
            uint16 EventID,
            uint16 EventType )
```

Definition at line 180 of file cfe_evs_utils.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_ES_GetAppName(), CFE_EVS_CRITICAL_BIT, CFE_EVS_DEBUG_BIT, CFE_EVS_ERROR_BIT, CFE_EVS_EventType_CRITICAL, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_FILTER_MAX_EID, CFE_EVS_GlobalData, CFE_EVS_INFORMATION_BIT, CFE_EVS_MAX_FILTER_COUNT, EVS_BinFilter_t::Count, EVS_AppData_t::EventTypesActiveFlag, EVS_FindEventID(), EVS_SendEvent(), EVS_BinFilter_t::Mask, NULL, and OS_MAX_API_NAME.

Referenced by CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), and EVS_SendEvent().

Here is the call graph for this function:



13.44.1.8 EVS_NotRegistered()

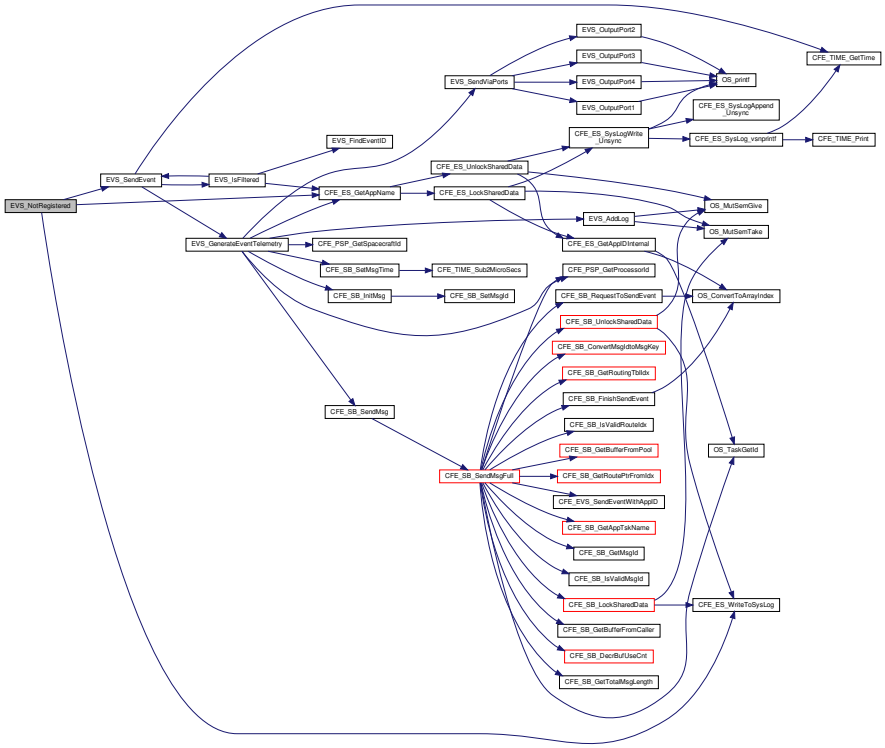
```
int32 EVS_NotRegistered (
            uint32 AppID )
```

Definition at line 139 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_ES_GetAppName(), CFE_ES_WriteToSysLog(), CFE_EVS_A↩
PP_NOT_REGISTERED, CFE_EVS_ERR_UNREGISTERED_EVS_APP, CFE_EVS_EventType_ERROR, CFE_EV↩
S_GlobalData, EVS_AppData_t::EventCount, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, OS_MAX↩
_API_NAME, CFE_EVS_HousekeepingTlm_t::Payload, and CFE_EVS_HousekeepingTlm_Payload_t::Unregistered↩
AppCounter.

Referenced by CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), and CFE_EVS_SendTimedEvent().

Here is the call graph for this function:



13.44.1.9 EVS_OutputPort1()

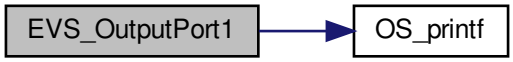
```
void EVS_OutputPort1 (
            char * Message )
```

Definition at line 507 of file cfe_evs_utils.c.

References OS_printf().

Referenced by EVS_SendViaPorts().

Here is the call graph for this function:



13.44.1.10 EVS_OutputPort2()

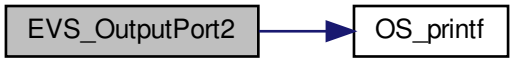
```
void EVS_OutputPort2 (
             char * Message )
```

Definition at line 525 of file cfe_evs_utils.c.

References OS_printf().

Referenced by EVS_SendViaPorts().

Here is the call graph for this function:



13.44.1.11 EVS_OutputPort3()

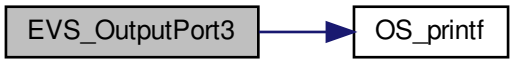
```
void EVS_OutputPort3 (
             char * Message )
```

Definition at line 543 of file cfe_evs_utils.c.

References OS_printf().

Referenced by EVS_SendViaPorts().

Here is the call graph for this function:



13.44.1.12 EVS_OutputPort4()

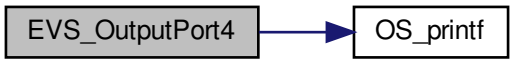
```
void EVS_OutputPort4 (
             char * Message )
```

Definition at line 561 of file cfe_evs_utils.c.

References OS_printf().

Referenced by EVS_SendViaPorts().

Here is the call graph for this function:



13.44.1.13 EVS_SendEvent()

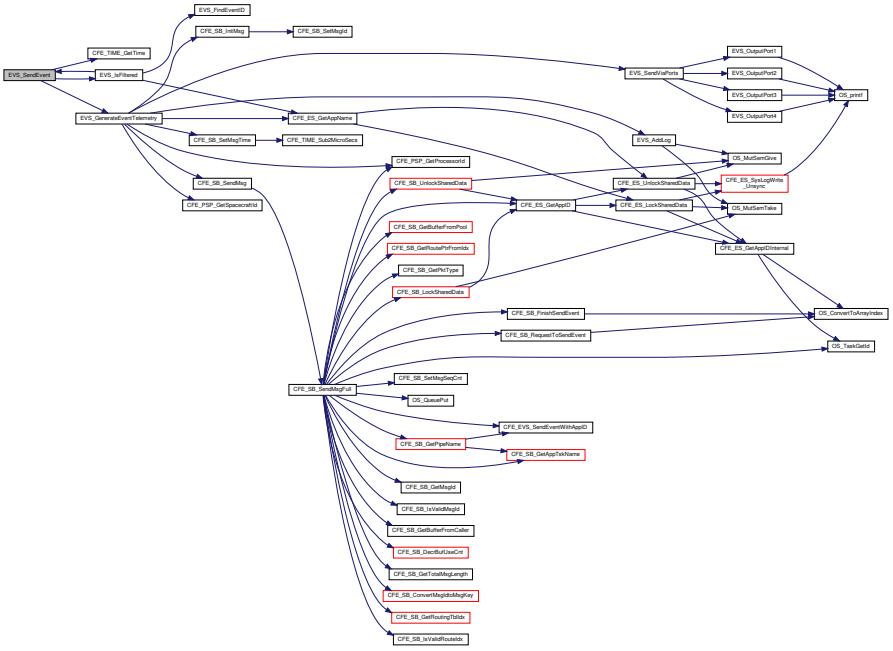
```
int32 EVS_SendEvent (
             uint16 EventID,
             uint16 EventType,
             const char * Spec,
              ...  )
```

Definition at line 580 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_TIME_↩ GetTime(), CFE_EVS_GlobalData_t::EVS_AppID, EVS_GenerateEventTelemetry(), and EVS_IsFiltered().

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_ClearLogCmd(), CFE_EVS_DeleteEventFilterCmd(), C↩ FE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_DisableEventTypeCmd(), C↩ FE_EVS_DisablePortsCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_E↩ VS_EnableEventTypeCmd(), CFE_EVS_EnablePortsCmd(), CFE_EVS_NoopCmd(), CFE_EVS_ProcessCommand↩ Packet(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounter↩ Cmd(), CFE_EVS_ResetCountersCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetEventFormatModeCmd(), CF↩ E_EVS_SetFilterCmd(), CFE_EVS_SetLogModeCmd(), CFE_EVS_TaskInit(), CFE_EVS_VerifyCmdLength(), CFE_↩ EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), EVS_IsFiltered(), and EVS_NotRegistered().

Here is the call graph for this function:



13.44.1.14 EVS_SendViaPorts()

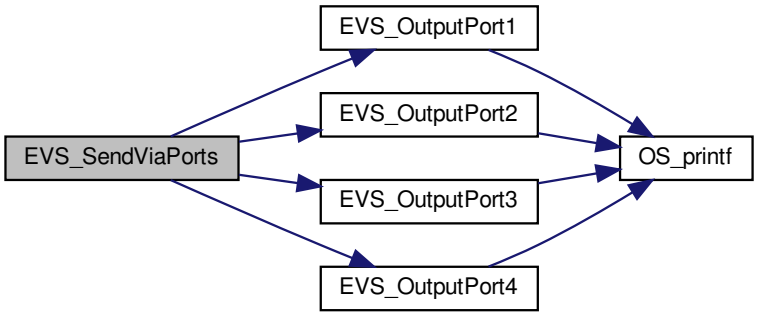
```
void EVS_SendViaPorts (
            CFE_EVS_LongEventTlm_t * EVS_PktPtr )
```

Definition at line 441 of file cfe_evs_utils.c.

References CFE_EVS_PacketID_t::AppName, CFE_EVS_GlobalData, CFE_EVS_MAX_PORT_MSG_LENGTH, CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT, CFE_EVS_PORT4_BIT, CFE_EVS_PacketID_t::EventID, EVS_OutputPort1(), EVS_OutputPort2(), EVS_OutputPort3(), EVS_OutputPort4(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_LongEventTlm_Payload_t::Message, CFE_EVS_HousekeepingTlm_Payload_t::OutputPort, CFE_EVS_LongEventTlm_Payload_t::PacketID, CFE_EVS_HousekeepingTlm_t::Payload, CFE_EVS_LongEventTlm_t::Payload, CFE_EVS_PacketID_t::ProcessorID, and CFE_EVS_PacketID_t::SpacecraftID.

Referenced by EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.45 cfe/fsw/cfe-core/src/evs/cfe_evs_utils.h File Reference

```
#include "cfe_evs_task.h"
```

Functions

- int32 EVS_GetAppID (uint32 *AppIdPtr)
- int32 EVS_GetApplicationInfo (uint32 *pAppID, const char *pAppName)
- int32 EVS_NotRegistered (uint32 AppID)
- bool EVS_IsFiltered (uint32 AppID, uint16 EventID, uint16 EventType)
- EVS_BinFilter_t * EVS_FindEventID (int16 EventID, EVS_BinFilter_t *FilterArray)
- void EVS_EnableTypes (uint8 BitMask, uint32 AppID)
- void EVS_DisableTypes (uint8 BitMask, uint32 AppID)
- void EVS_GenerateEventTelemetry (uint32 AppID, uint16 EventID, uint16 EventType, const CFE_TIME_Sys↩ Time_t *Time, const char *MsgSpec, va_list ArgPtr)
- int32 EVS_SendEvent (uint16 EventID, uint16 EventType, const char *Spec,...)

13.45.1 Function Documentation

13.45.1.1 EVS_DisableTypes()

```
void EVS_DisableTypes (
            uint8 BitMask,
            uint32 AppID )
```

Definition at line 336 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_CRITICAL_BIT, CFE_EVS_DEBUG_BIT, CFE_EVS_ER↩ ROR_BIT, CFE_EVS_GlobalData, CFE_EVS_INFORMATION_BIT, and EVS_AppData_t::EventTypesActiveFlag.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), and CFE_EVS_DisableEventTypeCmd().

13.45.1.2 EVS_EnableTypes()

```
void EVS_EnableTypes (
            uint8 BitMask,
            uint32 AppID )
```

Definition at line 316 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_CRITICAL_BIT, CFE_EVS_DEBUG_BIT, CFE_EVS_ER↩ ROR_BIT, CFE_EVS_GlobalData, CFE_EVS_INFORMATION_BIT, and EVS_AppData_t::EventTypesActiveFlag.

Referenced by CFE_EVS_EnableAppEventTypeCmd(), and CFE_EVS_EnableEventTypeCmd().

13.45.1.3 EVS_FindEventID()

```
EVS_BinFilter_t* EVS_FindEventID (
            int16 EventID,
            EVS_BinFilter_t * FilterArray )
```

Definition at line 289 of file cfe_evs_utils.c.

References CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, and NULL.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_ResetFilter(), CFE_↩ EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), and EVS_IsFiltered().

13.45.1.4 EVS_GenerateEventTelemetry()

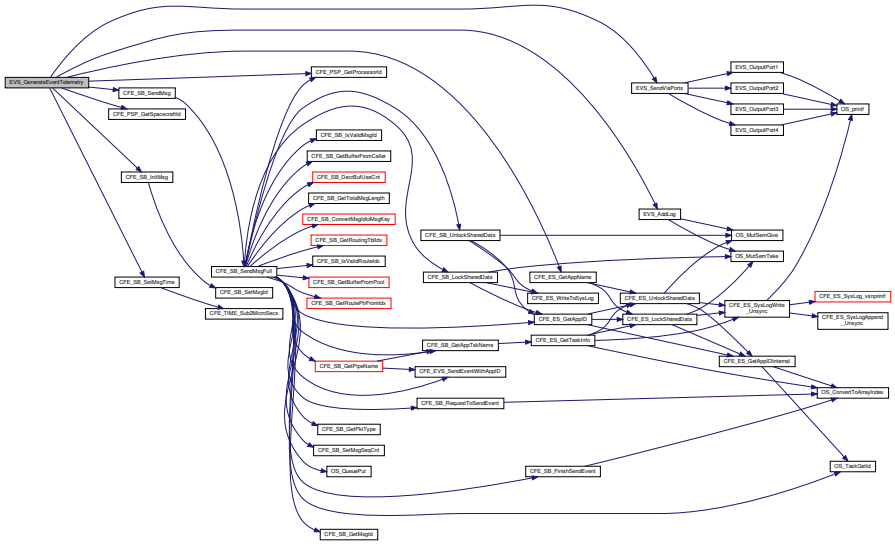
```
void EVS_GenerateEventTelemetry (
            uint32 AppID,
            uint16 EventID,
            uint16 EventType,
            const CFE_TIME_SysTime_t * Time,
            const char * MsgSpec,
            va_list ArgPtr )
```

Definition at line 360 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_PacketID_t::AppName, CFE_ES_GetAppName(), CFE↩ _EVS_GlobalData, CFE_EVS_LONG_EVENT_MSG_MID, CFE_EVS_MAX_EVENT_SEND_COUNT, CFE_EVS_↩ MSG_TRUNCATED, CFE_EVS_MsgFormat_LONG, CFE_EVS_MsgFormat_SHORT, CFE_EVS_SHORT_EVENT↩ _MSG_MID, CFE_PSP_GetProcessorId(), CFE_PSP_GetSpacecraftId(), CFE_SB_InitMsg(), CFE_SB_SendMsg(), CFE_SB_SetMsgTime(), EVS_AppData_t::EventCount, CFE_EVS_PacketID_t::EventID, CFE_EVS_PacketID_t::↩ EventType, EVS_AddLog(), EVS_SendViaPorts(), CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_LongEventTlm↩ _Payload_t::Message, CFE_EVS_HousekeepingTlm_Payload_t::MessageFormatMode, CFE_EVS_Housekeeping↩ Tlm_Payload_t::MessageSendCounter, CFE_EVS_HousekeepingTlm_Payload_t::MessageTruncCounter, CFE_EVS↩ _LongEventTlm_Payload_t::PacketID, CFE_EVS_HousekeepingTlm_t::Payload, CFE_EVS_LongEventTlm_t::Payload, CFE_EVS_PacketID_t::ProcessorID, and CFE_EVS_PacketID_t::SpacecraftID.

Referenced by CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), and EV↩
S_SendEvent().

Here is the call graph for this function:



13.45.1.5 EVS_GetAppID()

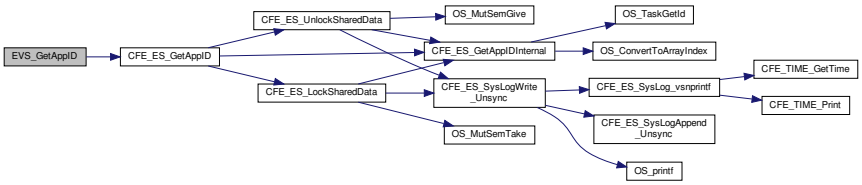
```
int32 EVS_GetAppID (
            uint32 * AppIdPtr )
```

Definition at line 67 of file cfe_evs_utils.c.

References CFE_ES_GetAppID(), CFE_EVS_APP_ILLEGAL_APP_ID, CFE_PLATFORM_ES_MAX_APPLICATIONS, and CFE_SUCCESS.

Referenced by CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetFilter(), CFE_EVS_SendEvent(), CFE_EVS_SendTimedEvent(), CFE_EVS_TaskInit(), and CFE_EVS_Unregister().

Here is the call graph for this function:



13.45.1.6 EVS_GetApplicationInfo()

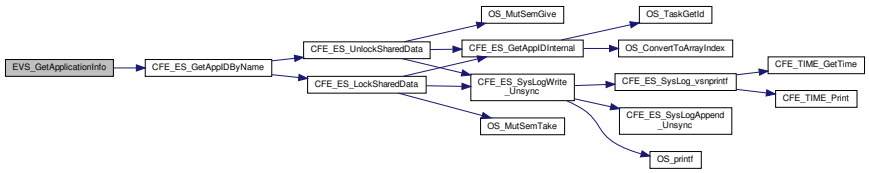
```
int32 EVS_GetApplicationInfo (
            uint32 * pAppID,
            const char * pAppName )
```

Definition at line 99 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_ES_ERR_BUFFER, CFE_ES_GetAppIDByName(), CFE_EVS↩_APP_ILLEGAL_APP_ID, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_GlobalData, CFE_PLATFORM_ES_M↩AX_APPLICATIONS, CFE_SUCCESS, NULL, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEvents↩Cmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEvent↩TypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

Here is the call graph for this function:



13.45.1.7 EVS_IsFiltered()

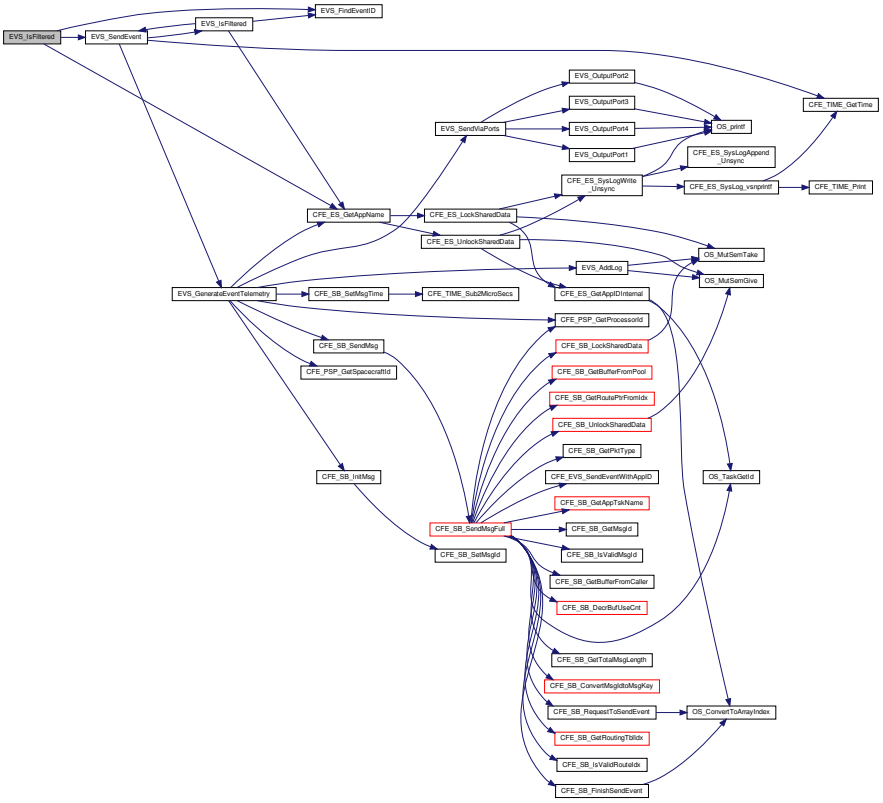
```
bool EVS_IsFiltered (
            uint32 AppID,
            uint16 EventID,
            uint16 EventType )
```

Definition at line 180 of file cfe_evs_utils.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_E↩S_GetAppName(), CFE_EVS_CRITICAL_BIT, CFE_EVS_DEBUG_BIT, CFE_EVS_ERROR_BIT, CFE_EVS_Event↩Type_CRITICAL, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORM↩ATION, CFE_EVS_FILTER_MAX_EID, CFE_EVS_GlobalData, CFE_EVS_INFORMATION_BIT, CFE_EVS_MAX_↩FILTER_COUNT, EVS_BinFilter_t::Count, EVS_AppData_t::EventTypesActiveFlag, EVS_FindEventID(), EVS_Send↩Event(), EVS_BinFilter_t::Mask, NULL, and OS_MAX_API_NAME.

Referenced by CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), and EV↩S_SendEvent().

Here is the call graph for this function:



13.45.1.8 EVS_NotRegistered()

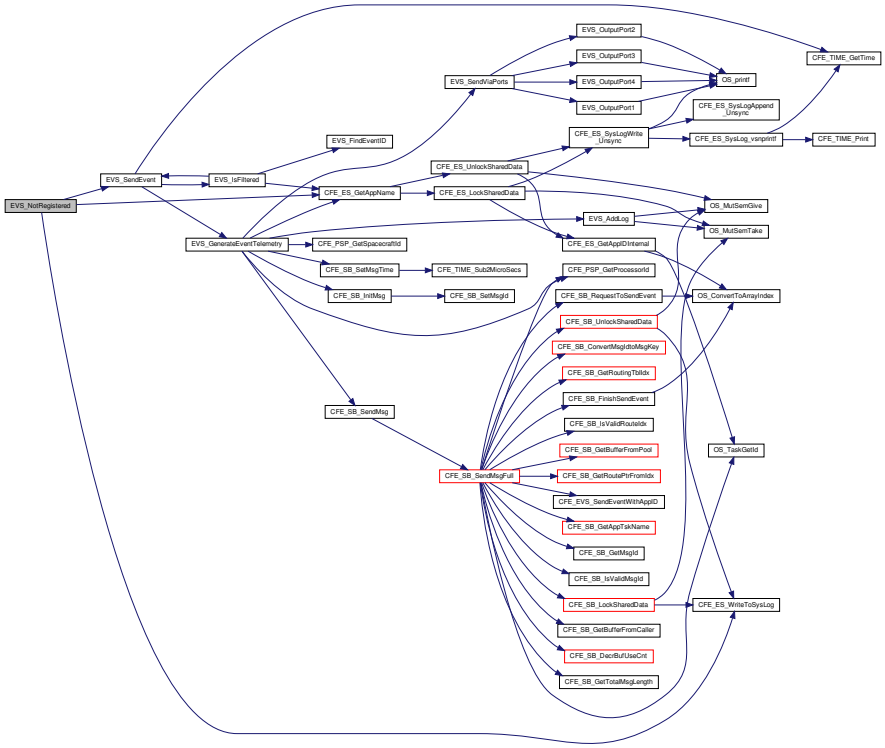
```
int32 EVS_NotRegistered (
            uint32 AppID )
```

Definition at line 139 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData_t::AppData, CFE_ES_GetAppName(), CFE_ES_WriteToSysLog(), CFE_EVS_A↩PP_NOT_REGISTERED, CFE_EVS_ERR_UNREGISTERED_EVS_APP, CFE_EVS_EventType_ERROR, CFE_EV↩S_GlobalData, EVS_AppData_t::EventCount, EVS_SendEvent(), CFE_EVS_GlobalData_t::EVS_TlmPkt, OS_MAX↩_API_NAME, CFE_EVS_HousekeepingTlm_t::Payload, and CFE_EVS_HousekeepingTlm_Payload_t::Unregistered↩AppCounter.

Referenced by CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), and CFE_EVS_SendTimedEvent().

Here is the call graph for this function:



13.45.1.9 EVS_SendEvent()

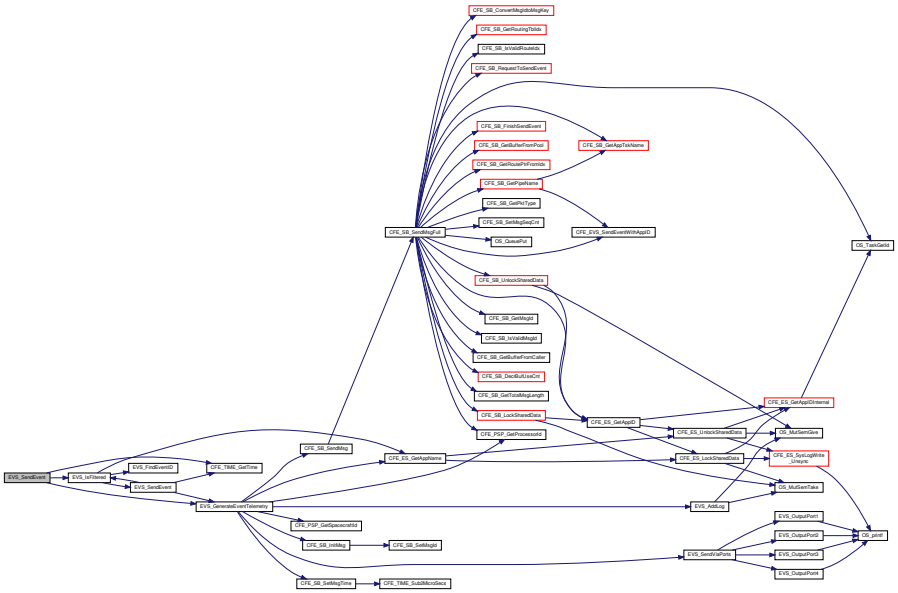
```
int32 EVS_SendEvent (
            uint16 EventID,
            uint16 EventType,
            const char * Spec,
             ...  )
```

Definition at line 580 of file cfe_evs_utils.c.

References CFE_EVS_GlobalData, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_TIME_↩ GetTime(), CFE_EVS_GlobalData_t::EVS_AppID, EVS_GenerateEventTelemetry(), and EVS_IsFiltered().

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_ClearLogCmd(), CFE_EVS_DeleteEventFilterCmd(), C↩ FE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_DisableEventTypeCmd(), C↩ FE_EVS_DisablePortsCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_E↩ VS_EnableEventTypeCmd(), CFE_EVS_EnablePortsCmd(), CFE_EVS_NoopCmd(), CFE_EVS_ProcessCommand↩ Packet(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounter↩ Cmd(), CFE_EVS_ResetCountersCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetEventFormatModeCmd(), CF↩ E_EVS_SetFilterCmd(), CFE_EVS_SetLogModeCmd(), CFE_EVS_TaskInit(), CFE_EVS_VerifyCmdLength(), CFE_↩ EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), EVS_IsFiltered(), and EVS_NotRegistered().

Here is the call graph for this function:



13.46 cfe/fsw/cfe-core/src/evs/cfe_evs_verify.h File Reference

13.47 cfe/fsw/cfe-core/src/inc/ccsds.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
```

Data Structures

- struct CCSDS_PriHdr_t
- struct CCSDS_CmdSecHdr_t
- struct CCSDS_TlmSecHdr_t
- struct CCSDS_APIDqualifiers_t
- struct CCSDS_APIDQHdr_t

  CCSDS Primary with APID Qualifier Header Type Definition.
- struct CCSDS_SpacePacket_t
- struct CCSDS_CommandPacket_t
- struct CCSDS_TelemetryPacket_t

Macros

- #define CFE_MAKE_BIG16(n) ((((n) << 8) & 0xFF00) | (((n) >> 8) & 0x00FF))
- #define CFE_MAKE_BIG32(n) ((((n) << 24) & 0xFF000000) | (((n) << 8) & 0x00FF0000) | (((n) >> 8) & 0x0000FF00) | (((n) >> 24) & 0x000000FF))
- #define CCSDS_TIME_SIZE 6
- #define CCSDS_BIG_ENDIAN 0
- #define CCSDS_LITTLE_ENDIAN 1
- #define CCSDS_ENDIAN_MASK 0x0400
- #define CCSDS_NON_PLAYBACK_PKT 0
- #define CCSDS_PLAYBACK_PKT 1
- #define CCSDS_PLAYBACK_PKT_MASK 0x0200
- #define CCSDS_EDS_MASK 0xF800
- #define CCSDS_TLM 0
- #define CCSDS_CMD 1
- #define CCSDS_NO_SEC_HDR 0
- #define CCSDS_HAS_SEC_HDR 1
- #define NUM_CCSDS_APIDS 2048
- #define NUM_CCSDS_PKT_TYPES 2
- #define CCSDS_INIT_SEQ 0
- #define CCSDS_INIT_SEQFLG 3
- #define CCSDS_INIT_FC 0
- #define CCSDS_INIT_CHECKSUM 0
- #define CCSDS_RD_BITS(word, mask, shift) (((word) & mask) >> shift)
- #define CCSDS_WR_BITS(word, mask, shift, value) ((word) = (uint16)(((word) & ~mask) | (((value) & (mask >> shift)) << shift)))
- #define CCSDS_RD_SID(phdr) (((phdr).StreamId[0] << 8) + ((phdr).StreamId[1]))
- #define CCSDS_WR_SID(phdr, value)
- #define CCSDS_RD_APID(phdr) (CCSDS_RD_SID(phdr) & 0x07FF)
- #define CCSDS_WR_APID(phdr, value)
- #define CCSDS_RD_SHDR(phdr) (((phdr).StreamId[0] & 0x08) >> 3)
- #define CCSDS_WR_SHDR(phdr, value) ((phdr).StreamId[0] = ((phdr).StreamId[0] & 0xf7) | ((value << 3) & 0x08))
- #define CCSDS_RD_TYPE(phdr) (((phdr).StreamId[0] & 0x10) >> 4)
- #define CCSDS_WR_TYPE(phdr, value) ((phdr).StreamId[0] = ((phdr).StreamId[0] & 0xEF) | ((value << 4) & 0x10))
- #define CCSDS_RD_VERS(phdr) (((phdr).StreamId[0] & 0xE0) >> 5)
- #define CCSDS_WR_VERS(phdr, value) ((phdr).StreamId[0] = ((phdr).StreamId[0] & 0x1F) | ((value << 5) & 0xE0))
- #define CCSDS_RD_SEQ(phdr) ((((phdr).Sequence[0] & 0x3F) << 8) + ((phdr).Sequence[1]))
- #define CCSDS_WR_SEQ(phdr, value)
- #define CCSDS_RD_SEQFLG(phdr) (((phdr).Sequence[0] & 0xC0) >> 6)
- #define CCSDS_WR_SEQFLG(phdr, value) ((phdr).Sequence[0] = ((phdr).Sequence[0] & 0x3F) | ((value << 6) & 0xC0))
- #define CCSDS_RD_LEN(phdr) (((phdr).Length[0] << 8) + (phdr).Length[1] + 7)
- #define CCSDS_WR_LEN(phdr, value)
- #define CCSDS_RD_FC(shdr) CCSDS_RD_BITS((shdr).Command, 0x7F00, 8)
- #define CCSDS_WR_FC(shdr, value) CCSDS_WR_BITS((shdr).Command, 0x7F00, 8, value)
- #define CCSDS_RD_CHECKSUM(shdr) CCSDS_RD_BITS((shdr).Command, 0x00FF, 0)
- #define CCSDS_WR_CHECKSUM(shdr, val) CCSDS_WR_BITS((shdr).Command, 0x00FF, 0, val)
- #define CCSDS_RD_EDS_VER(shdr) (((shdr).APIDQSubsystem[0] & 0xF8) >> 3)

- #define CCSDS_RD_ENDIAN(shdr) (((shdr).APIDQSubsystem[0] & 0x04) >> 2)
- #define CCSDS_RD_PLAYBACK(shdr) (((shdr).APIDQSubsystem[0] & 0x02) >> 1)
- #define CCSDS_RD_SUBSYSTEM_ID(shdr) ((((shdr).APIDQSubsystem[0] & 0x01) << 8) + ((shdr).APIDQ↩ Subsystem[1]))
- #define CCSDS_RD_SYSTEM_ID(shdr) (((shdr).APIDQSystemId[0] << 8) + ((shdr).APIDQSystemId[1]))
- #define CCSDS_WR_EDS_VER(shdr, val) ((shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0x07) | (((val) & 0x1f) << 3))
- #define CCSDS_WR_ENDIAN(shdr, val) ((shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0xFB) | (((val) & 0x01) << 2))
- #define CCSDS_WR_PLAYBACK(shdr, val) ((shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0xFD) | (((val) & 0x01) << 1))
- #define CCSDS_WR_SUBSYSTEM_ID(shdr, val)
- #define CCSDS_WR_SYSTEM_ID(shdr, val)
- #define CCSDS_CLR_PRI_HDR(phdr)
- #define CCSDS_CLR_SEC_APIDQ(shdr)
- #define CCSDS_CLR_CMDSEC_HDR(shdr) ((shdr).Command = (CCSDS_INIT_CHECKSUM << 0) | (CCS↩ DS_INIT_FC << 8))
- #define CCSDS_WR_SEC_HDR_SEC(shdr, value)
- #define CCSDS_RD_SEC_HDR_SEC(shdr)
- #define CCSDS_CLR_TLMSEC_HDR(shdr)
- #define CCSDS_WR_SEC_HDR_SUBSEC(shdr, value)
- #define CCSDS_RD_SEC_HDR_SUBSEC(shdr)
- #define CCSDS_SID_APID(sid) CCSDS_RD_BITS(sid, 0x07FF, 0)
- #define CCSDS_SID_SHDR(sid) CCSDS_RD_BITS(sid, 0x0800, 11)
- #define CCSDS_SID_TYPE(sid) CCSDS_RD_BITS(sid, 0x1000, 12)
- #define CCSDS_SID_VERS(sid) CCSDS_RD_BITS(sid, 0xE000, 13)
- #define CCSDS_INC_SEQ(phdr) CCSDS_WR_SEQ(phdr, (CCSDS_RD_SEQ(phdr)+1))

Typedefs

- typedef CCSDS_CommandPacket_t CCSDS_CmdPkt_t
- typedef CCSDS_TelemetryPacket_t CCSDS_TlmPkt_t

Functions

- void CCSDS_LoadCheckSum (CCSDS_CommandPacket_t *PktPtr)
- bool CCSDS_ValidCheckSum (CCSDS_CommandPacket_t *PktPtr)
- uint8 CCSDS_ComputeCheckSum (CCSDS_CommandPacket_t *PktPtr)

13.47.1 Macro Definition Documentation

13.47.1.1 CCSDS_BIG_ENDIAN

```
#define CCSDS_BIG_ENDIAN 0
```

Definition at line 127 of file ccsds.h.

13.47.1.2 CCSDS_CLR_CMDSEC_HDR

```
#define CCSDS_CLR_CMDSEC_HDR(
              shdr ) ( (shdr).Command = (CCSDS_INIT_CHECKSUM << 0) | (CCSDS_INIT_FC << 8) )
```

Definition at line 380 of file ccsds.h.

13.47.1.3 CCSDS_CLR_PRI_HDR

```
#define CCSDS_CLR_PRI_HDR(
              phdr )
```

Value:

```
( (phdr).StreamId[0] = 0,\
    (phdr).StreamId[1] = 0,\
    (phdr).Sequence[0] = (CCSDS_INIT_SEQFLG << 6),\
    (phdr).Sequence[1] = 0,\
    (phdr).Length[0] = 0, \
    (phdr).Length[1] = 0 )
```

Definition at line 365 of file ccsds.h.

Referenced by CFE_SB_InitMsg().

13.47.1.4 CCSDS_CLR_SEC_APIDQ

```
#define CCSDS_CLR_SEC_APIDQ(
              shdr )
```

Value:

```
( (shdr).APIDQSubsystem[0] = 0,\
    (shdr).APIDQSubsystem[1] = 0,\
    (shdr).APIDQSystemId[0] = 0,\
    (shdr).APIDQSystemId[1] = 0 )
```

Definition at line 373 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.5 CCSDS_CLR_TLMSEC_HDR

```
#define CCSDS_CLR_TLMSEC_HDR(
              shdr )
```

Value:

```
( (shdr).Time[0] = 0,\
    (shdr).Time[1] = 0,\
    (shdr).Time[2] = 0,\
    (shdr).Time[3] = 0,\
    (shdr).Time[4] = 0,\
    (shdr).Time[5] = 0 )
```

Definition at line 397 of file ccsds.h.

13.47.1.6 CCSDS_CMD

```
#define CCSDS_CMD 1
```

Definition at line 226 of file ccsds.h.

Referenced by CFE_SB_GetMsgId(), CFE_SB_GetMsgTime(), CFE_SB_MsgHdrSize(), and CFE_SB_SetMsgTime().

13.47.1.7 CCSDS_EDS_MASK

```
#define CCSDS_EDS_MASK 0xF800
```

Definition at line 140 of file ccsds.h.

13.47.1.8 CCSDS_ENDIAN_MASK

```
#define CCSDS_ENDIAN_MASK 0x0400
```

Definition at line 129 of file ccsds.h.

13.47.1.9 CCSDS_HAS_SEC_HDR

```
#define CCSDS_HAS_SEC_HDR 1
```

Definition at line 231 of file ccsds.h.

13.47.1.10 CCSDS_INC_SEQ

```
#define CCSDS_INC_SEQ(
             phdr ) CCSDS_WR_SEQ(phdr, (CCSDS_RD_SEQ(phdr)+1))
```

Definition at line 464 of file ccsds.h.

13.47.1.11 CCSDS_INIT_CHECKSUM

```
#define CCSDS_INIT_CHECKSUM 0
```

Definition at line 248 of file ccsds.h.

13.47.1.12 CCSDS_INIT_FC

```
#define CCSDS_INIT_FC 0
```

Definition at line 246 of file ccsds.h.

13.47.1.13 CCSDS_INIT_SEQ

```
#define CCSDS_INIT_SEQ 0
```

Definition at line 242 of file ccsds.h.

13.47.1.14 CCSDS_INIT_SEQFLG

```
#define CCSDS_INIT_SEQFLG 3
```

Definition at line 244 of file ccsds.h.

Referenced by CFE_SB_InitMsg().

13.47.1.15 CCSDS_LITTLE_ENDIAN

```
#define CCSDS_LITTLE_ENDIAN 1
```

Definition at line 128 of file ccsds.h.

13.47.1.16 CCSDS_NO_SEC_HDR

```
#define CCSDS_NO_SEC_HDR 0
```

Definition at line 229 of file ccsds.h.

13.47.1.17 CCSDS_NON_PLAYBACK_PKT

```
#define CCSDS_NON_PLAYBACK_PKT 0
```

Definition at line 133 of file ccsds.h.

13.47.1.18 CCSDS_PLAYBACK_PKT

```
#define CCSDS_PLAYBACK_PKT 1
```

Definition at line 134 of file ccsds.h.

13.47.1.19 CCSDS_PLAYBACK_PKT_MASK

```
#define CCSDS_PLAYBACK_PKT_MASK 0x0200
```

Definition at line 135 of file ccsds.h.

13.47.1.20 CCSDS_RD_APID

```
#define CCSDS_RD_APID(
              phdr ) (CCSDS_RD_SID(phdr) & 0x07FF)
```

Definition at line 291 of file ccsds.h.

Referenced by CFE_SB_GetMsgId().

13.47.1.21 CCSDS_RD_BITS

```
#define CCSDS_RD_BITS(
              word,
              mask,
              shift ) (((word) & mask) >> shift)
```

Definition at line 260 of file ccsds.h.

13.47.1.22 CCSDS_RD_CHECKSUM

```
#define CCSDS_RD_CHECKSUM(
              shdr ) CCSDS_RD_BITS((shdr).Command, 0x00FF, 0)
```

Definition at line 334 of file ccsds.h.

Referenced by CFE_SB_GetChecksum().

13.47.1.23 CCSDS_RD_EDS_VER

```
#define CCSDS_RD_EDS_VER(
              shdr ) ( ((shdr).APIDQSubsystem[0] & 0xF8) >> 3)
```

Definition at line 342 of file ccsds.h.

13.47.1.24 CCSDS_RD_ENDIAN

```
#define CCSDS_RD_ENDIAN(
              shdr ) ( ((shdr).APIDQSubsystem[0] & 0x04) >> 2)
```

Definition at line 343 of file ccsds.h.

13.47.1.25 CCSDS_RD_FC

```
#define CCSDS_RD_FC(
              shdr ) CCSDS_RD_BITS((shdr).Command, 0x7F00, 8)
```

Definition at line 329 of file ccsds.h.

Referenced by CFE_SB_GetCmdCode().

13.47.1.26 CCSDS_RD_LEN

```
#define CCSDS_RD_LEN(
              phdr ) ( ( (phdr).Length[0] << 8) + (phdr).Length[1] + 7)
```

Definition at line 323 of file ccsds.h.

Referenced by CCSDS_ComputeCheckSum(), and CFE_SB_GetTotalMsgLength().

13.47.1.27 CCSDS_RD_PLAYBACK

```
#define CCSDS_RD_PLAYBACK(
              shdr ) ( ((shdr).APIDQSubsystem[0] & 0x02) >> 1)
```

Definition at line 344 of file ccsds.h.

13.47.1.28 CCSDS_RD_SEC_HDR_SEC

```
#define CCSDS_RD_SEC_HDR_SEC(
              shdr )
```

Value:

```
(((uint32)shdr.Time[0]) << 24) | \
                                        (((uint32)shdr.Time[1]) << 16) | \
                                        (((uint32)shdr.Time[2]) << 8)  | \
                                        ((uint32)shdr.Time[3])
```

Definition at line 389 of file ccsds.h.

13.47.1.29 CCSDS_RD_SEC_HDR_SUBSEC

```
#define CCSDS_RD_SEC_HDR_SUBSEC(
              shdr )
```

Value:

```
(((uint32)shdr.Time[4]) << 8)  | \
                                        ((uint32)shdr.Time[5])
```

Definition at line 409 of file ccsds.h.

13.47.1.30 CCSDS_RD_SEQ

```
#define CCSDS_RD_SEQ(
              phdr ) ((((phdr).Sequence[0] & 0x3F) << 8) + ((phdr).Sequence[1]))
```

Definition at line 312 of file ccsds.h.

Referenced by CFE_SB_InitMsg().

13.47.1.31 CCSDS_RD_SEQFLG

```
#define CCSDS_RD_SEQFLG(
              phdr ) (((phdr).Sequence[0] & 0xC0) >> 6)
```

Definition at line 318 of file ccsds.h.

13.47.1.32 CCSDS_RD_SHDR

```
#define CCSDS_RD_SHDR(
              phdr ) (((phdr).StreamId[0] & 0x08) >> 3)
```

Definition at line 297 of file ccsds.h.

Referenced by CFE_SB_GenerateChecksum(), CFE_SB_GetChecksum(), CFE_SB_GetCmdCode(), CFE_SB_↩ GetMsgTime(), CFE_SB_MsgHdrSize(), CFE_SB_SetCmdCode(), CFE_SB_SetMsgTime(), and CFE_SB_Validate↩ Checksum().

13.47.1.33 CCSDS_RD_SID

```
#define CCSDS_RD_SID(
              phdr ) (((phdr).StreamId[0] << 8) + ((phdr).StreamId[1]))
```

Definition at line 285 of file ccsds.h.

Referenced by CFE_SB_GetMsgId().

13.47.1.34 CCSDS_RD_SUBSYSTEM_ID

```
#define CCSDS_RD_SUBSYSTEM_ID(
              shdr ) ( (((shdr).APIDQSubsystem[0] & 0x01) << 8) + ((shdr).APIDQSubsystem[1]))
```

Definition at line 345 of file ccsds.h.

Referenced by CFE_SB_GetMsgId().

13.47.1.35 CCSDS_RD_SYSTEM_ID

```
#define CCSDS_RD_SYSTEM_ID(
              shdr ) ( ((shdr).APIDQSystemId[0] << 8) + ((shdr).APIDQSystemId[1]))
```

Definition at line 346 of file ccsds.h.

13.47.1.36 CCSDS_RD_TYPE

```
#define CCSDS_RD_TYPE(
              phdr ) (((phdr).StreamId[0] & 0x10) >> 4)
```

Definition at line 302 of file ccsds.h.

Referenced by CFE_SB_GenerateChecksum(), CFE_SB_GetChecksum(), CFE_SB_GetCmdCode(), CFE_SB_Get↩MsgId(), CFE_SB_GetMsgTime(), CFE_SB_MsgHdrSize(), CFE_SB_SetCmdCode(), CFE_SB_SetMsgTime(), and CFE_SB_ValidateChecksum().

13.47.1.37 CCSDS_RD_VERS

```
#define CCSDS_RD_VERS(
              phdr ) (((phdr).StreamId[0] & 0xE0) >> 5)
```

Definition at line 307 of file ccsds.h.

13.47.1.38 CCSDS_SID_APID

```
#define CCSDS_SID_APID(
              sid ) CCSDS_RD_BITS(sid, 0x07FF, 0)
```

Definition at line 445 of file ccsds.h.

13.47.1.39 CCSDS_SID_SHDR

```
#define CCSDS_SID_SHDR(
              sid ) CCSDS_RD_BITS(sid, 0x0800, 11)
```

Definition at line 448 of file ccsds.h.

13.47.1.40 CCSDS_SID_TYPE

```
#define CCSDS_SID_TYPE(
              sid ) CCSDS_RD_BITS(sid, 0x1000, 12)
```

Definition at line 451 of file ccsds.h.

13.47.1.41 CCSDS_SID_VERS

```
#define CCSDS_SID_VERS(
              sid ) CCSDS_RD_BITS(sid, 0xE000, 13)
```

Definition at line 454 of file ccsds.h.

13.47.1.42 CCSDS_TIME_SIZE

```
#define CCSDS_TIME_SIZE 6
```

Definition at line 53 of file ccsds.h.

13.47.1.43 CCSDS_TLM

```
#define CCSDS_TLM 0
```

Definition at line 224 of file ccsds.h.

Referenced by CFE_SB_GenerateChecksum(), CFE_SB_GetChecksum(), CFE_SB_GetCmdCode(), CFE_SB_Set↩CmdCode(), and CFE_SB_ValidateChecksum().

13.47.1.44 CCSDS_WR_APID

```
#define CCSDS_WR_APID(
              phdr,
              value )
```

Value:

```
((((phdr).StreamId[0] = ((phdr).StreamId[0] & 0xF8) | ((value >> 8) & 0x07))) ,\
                                  (((phdr).StreamId[1]  = ((value)) & 0xff)) )
```

Definition at line 293 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.45 CCSDS_WR_BITS

```
#define CCSDS_WR_BITS(
              word,
              mask,
              shift,
              value ) ((word) = (uint16)(((word) & ~mask) | (((value) & (mask >> shift)) <<
shift)))
```

Definition at line 265 of file ccsds.h.

13.47.1.46 CCSDS_WR_CHECKSUM

```
#define CCSDS_WR_CHECKSUM(
              shdr,
              val ) CCSDS_WR_BITS((shdr).Command, 0x00FF, 0, val)
```

Definition at line 336 of file ccsds.h.

Referenced by CCSDS_LoadCheckSum().

13.47.1.47 CCSDS_WR_EDS_VER

```
#define CCSDS_WR_EDS_VER(
              shdr,
              val ) ( (shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0x07) | (((val) &
0x1f) << 3) )
```

Definition at line 348 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.48 CCSDS_WR_ENDIAN

```
#define CCSDS_WR_ENDIAN(
              shdr,
              val ) ( (shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0xFB) | (((val) &
0x01) << 2) )
```

Definition at line 349 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.49 CCSDS_WR_FC

```
#define CCSDS_WR_FC(
              shdr,
              value ) CCSDS_WR_BITS((shdr).Command, 0x7F00, 8, value)
```

Definition at line 331 of file ccsds.h.

Referenced by CFE_SB_SetCmdCode().

13.47.1.50 CCSDS_WR_LEN

```
#define CCSDS_WR_LEN(
              phdr,
              value )
```

Value:

```
((((phdr).Length[0] = ((value) - 7) >> 8)) ,\
                                ((phdr).Length[1] = ((value) - 7) & 0xff)) )
```

Definition at line 325 of file ccsds.h.

Referenced by CFE_SB_InitMsg(), CFE_SB_SetTotalMsgLength(), and CFE_SB_SetUserDataLength().

13.47.1.51 CCSDS_WR_PLAYBACK

```
#define CCSDS_WR_PLAYBACK(
              shdr,
              val ) ( (shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0xFD) | (((val) &
0x01) << 1) )
```

Definition at line 350 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.52 CCSDS_WR_SEC_HDR_SEC

```
#define CCSDS_WR_SEC_HDR_SEC(
              shdr,
              value )
```

Value:

```
shdr.Time[0] = ((value>>24) & 0xFF),  \
                                        shdr.Time[1] = ((value>>16) & 0xFF),  \
                                        shdr.Time[2] = ((value>>8)  & 0xFF),  \
                                        shdr.Time[3] = ((value)     & 0xFF)
```

Definition at line 384 of file ccsds.h.

13.47.1.53 CCSDS_WR_SEC_HDR_SUBSEC

```
#define CCSDS_WR_SEC_HDR_SUBSEC(
              shdr,
              value )
```

Value:

```
shdr.Time[4] = ((value>>8)  & 0xFF),  \
                                        shdr.Time[5] = ((value)     & 0xFF)
```

Definition at line 406 of file ccsds.h.

13.47.1.54 CCSDS_WR_SEQ

```
#define CCSDS_WR_SEQ(
              phdr,
              value )
```

Value:

```
((((phdr).Sequence[0] = ((phdr).Sequence[0] & 0xC0) | ((value >> 8) & 0x3f))) ,\
                                  (((phdr).Sequence[1]  = ((value)) & 0xff)) )
```

Definition at line 314 of file ccsds.h.

Referenced by CFE_SB_InitMsg(), and CFE_SB_SetMsgSeqCnt().

13.47.1.55 CCSDS_WR_SEQFLG

```
#define CCSDS_WR_SEQFLG(
              phdr,
              value ) ((phdr).Sequence[0] = ((phdr).Sequence[0] & 0x3F) | ((value << 6) & 0xC0) )
```

Definition at line 320 of file ccsds.h.

Referenced by CFE_SB_InitMsg().

13.47.1.56 CCSDS_WR_SHDR

```
#define CCSDS_WR_SHDR(
              phdr,
              value ) ((phdr).StreamId[0] = ((phdr).StreamId[0] & 0xf7) | ((value << 3) & 0x08))
```

Definition at line 299 of file ccsds.h.

Referenced by CFE_SB_InitMsg().

13.47.1.57 CCSDS_WR_SID

```
#define CCSDS_WR_SID(
              phdr,
              value )
```

Value:

```
( ((phdr).StreamId[0] = (value >> 8)   ) ,\
                                    ((phdr).StreamId[1] = (value & 0xff) ) )
```

Definition at line 287 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.58 CCSDS_WR_SUBSYSTEM_ID

```
#define CCSDS_WR_SUBSYSTEM_ID(
              shdr,
              val )
```

Value:

```
(((shdr).APIDQSubsystem[0] = ((shdr).APIDQSubsystem[0] & 0xFE) | ((val & 0x0100) >> 8)) ,\
                                      ( (shdr).APIDQSubsystem[1] =  (val & 0x00ff)) )
```

Definition at line 352 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.59 CCSDS_WR_SYSTEM_ID

```
#define CCSDS_WR_SYSTEM_ID(
              shdr,
              val )
```

Value:

```
(((shdr).APIDQSystemId[0] = ((val & 0xff00) >> 8)),\
                                      ( (shdr).APIDQSystemId[1] =  (val & 0x00ff)) )
```

Definition at line 355 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.60 CCSDS_WR_TYPE

```
#define CCSDS_WR_TYPE(
             phdr,
             value ) ((phdr).StreamId[0] = ((phdr).StreamId[0] & 0xEF) | ((value << 4) & 0x10))
```

Definition at line 304 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.61 CCSDS_WR_VERS

```
#define CCSDS_WR_VERS(
             phdr,
             value ) ((phdr).StreamId[0] = ((phdr).StreamId[0] & 0x1F) | ((value << 5) & 0xE0))
```

Definition at line 309 of file ccsds.h.

Referenced by CFE_SB_SetMsgId().

13.47.1.62 CFE_MAKE_BIG16

```
#define CFE_MAKE_BIG16(
             n ) ( (((n) << 8) & 0xFF00) | (((n) >> 8) & 0x00FF) )
```

Definition at line 45 of file ccsds.h.

13.47.1.63 CFE_MAKE_BIG32

```
#define CFE_MAKE_BIG32(
             n ) ( (((n) << 24) & 0xFF000000) | (((n) << 8) & 0x00FF0000) | (((n) >> 8) & 0x0000↩
FF00) | (((n) >> 24) & 0x000000FF) )
```

Definition at line 46 of file ccsds.h.

13.47.1.64 NUM_CCSDS_APIDS

```
#define NUM_CCSDS_APIDS 2048
```

Definition at line 233 of file ccsds.h.

13.47.1.65 NUM_CCSDS_PKT_TYPES

```
#define NUM_CCSDS_PKT_TYPES 2
```

Definition at line 234 of file ccsds.h.

13.47.2 Typedef Documentation

13.47.2.1 CCSDS_CmdPkt_t

```
typedef CCSDS_CommandPacket_t CCSDS_CmdPkt_t
```

Definition at line 210 of file ccsds.h.

13.47.2.2 CCSDS_TlmPkt_t

```
typedef CCSDS_TelemetryPacket_t CCSDS_TlmPkt_t
```

Definition at line 211 of file ccsds.h.

13.47.3 Function Documentation

13.47.3.1 CCSDS_ComputeCheckSum()

```
uint8 CCSDS_ComputeCheckSum (
            CCSDS_CommandPacket_t * PktPtr )
```

Definition at line 109 of file ccsds.c.

References CCSDS_RD_LEN, CCSDS_SpacePacket_t::Hdr, and CCSDS_CommandPacket_t::SpacePacket.

Referenced by CCSDS_LoadCheckSum(), and CCSDS_ValidCheckSum().

13.47.3.2 CCSDS_LoadCheckSum()

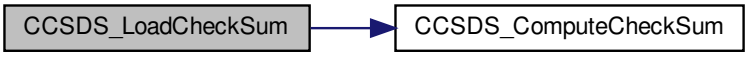
```
void CCSDS_LoadCheckSum (
            CCSDS_CommandPacket_t * PktPtr )
```

Definition at line 55 of file ccsds.c.

References CCSDS_ComputeCheckSum(), CCSDS_WR_CHECKSUM, and CCSDS_CommandPacket_t::Sec.

Referenced by CFE_SB_GenerateChecksum().

Here is the call graph for this function:



13.47.3.3 CCSDS_ValidCheckSum()

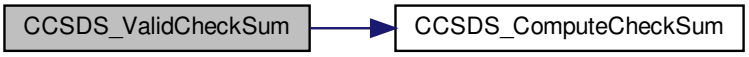
```
bool CCSDS_ValidCheckSum (
            CCSDS_CommandPacket_t * PktPtr )
```

Definition at line 85 of file ccsds.c.

References CCSDS_ComputeCheckSum().

Referenced by CFE_SB_ValidateChecksum().

Here is the call graph for this function:



13.48 cfe/fsw/cfe-core/src/inc/cfe.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_mission_cfg.h"
#include "cfe_error.h"
#include "cfe_es.h"
#include "cfe_evs.h"
#include "cfe_fs.h"
#include "cfe_sb.h"
#include "cfe_time.h"
#include "cfe_tbl.h"
#include "cfe_psp.h"
```

13.49 cfe/fsw/cfe-core/src/inc/cfe_error.h File Reference

```
#include "osapi.h"
```

Macros

- #define CFE_SEVERITY_BITMASK ((int32)0xc0000000)
- #define CFE_SEVERITY_SUCCESS ((int32)0x00000000)
- #define CFE_SEVERITY_INFO ((int32)0x40000000)
- #define CFE_SEVERITY_ERROR ((int32)0xc0000000)
- #define CFE_SERVICE_BITMASK ((int32)0x0e000000)
- #define CFE_EVENTS_SERVICE ((int32)0x02000000)
- #define CFE_EXECUTIVE_SERVICE ((int32)0x04000000)
- #define CFE_FILE_SERVICE ((int32)0x06000000)
- #define CFE_GENERIC_SERVICE ((int32)0x08000000)
- #define CFE_SOFTWARE_BUS_SERVICE ((int32)0x0a000000)
- #define CFE_TABLE_SERVICE ((int32)0x0c000000)
- #define CFE_TIME_SERVICE ((int32)0x0e000000)
- #define CFE_SUCCESS (0)
- #define CFE_STATUS_NO_COUNTER_INCREMENT ((int32)0x48000001)
- #define CFE_STATUS_WRONG_MSG_LENGTH ((int32)0xc8000002)
- #define CFE_STATUS_UNKNOWN_MSG_ID ((int32)0xc8000003)
- #define CFE_STATUS_BAD_COMMAND_CODE ((int32)0xc8000004)
- #define CFE_STATUS_NOT_IMPLEMENTED ((int32)0xc800ffff)
- #define CFE_EVS_UNKNOWN_FILTER ((int32)0xc2000001)
- #define CFE_EVS_APP_NOT_REGISTERED ((int32)0xc2000002)
- #define CFE_EVS_APP_ILLEGAL_APP_ID ((int32)0xc2000003)
- #define CFE_EVS_APP_FILTER_OVERLOAD ((int32)0xc2000004)
- #define CFE_EVS_RESET_AREA_POINTER ((int32)0xc2000005)
- #define CFE_EVS_EVT_NOT_REGISTERED ((int32)0xc2000006)
- #define CFE_EVS_FILE_WRITE_ERROR ((int32)0xc2000007)
- #define CFE_EVS_INVALID_PARAMETER ((int32)0xc2000008)
- #define CFE_EVS_FUNCTION_DISABLED ((int32)0xc2000009)

- #define CFE_EVS_NOT_IMPLEMENTED ((int32)0xc200ffff)
- #define CFE_ES_ERR_APPID ((int32)0xc4000001)
- #define CFE_ES_ERR_APPNAME ((int32)0xc4000002)
- #define CFE_ES_ERR_BUFFER ((int32)0xc4000003)
- #define CFE_ES_ERR_APP_CREATE ((int32)0xc4000004)
- #define CFE_ES_ERR_CHILD_TASK_CREATE ((int32)0xc4000005)
- #define CFE_ES_ERR_SYS_LOG_FULL ((int32)0xc4000006)
- #define CFE_ES_ERR_MEM_HANDLE ((int32)0xc4000007)
- #define CFE_ES_ERR_MEM_BLOCK_SIZE ((int32)0xc4000008)
- #define CFE_ES_ERR_LOAD_LIB ((int32)0xc4000009)
- #define CFE_ES_BAD_ARGUMENT ((int32)0xc400000a)
- #define CFE_ES_ERR_CHILD_TASK_REGISTER ((int32)0xc400000b)
- #define CFE_ES_ERR_SHELL_CMD ((int32)0xc400000c)
- #define CFE_ES_CDS_ALREADY_EXISTS ((int32)0x4400000d)
- #define CFE_ES_CDS_INSUFFICIENT_MEMORY ((int32)0xc400000e)
- #define CFE_ES_CDS_INVALID_NAME ((int32)0xc400000f)
- #define CFE_ES_CDS_INVALID_SIZE ((int32)0xc4000010)
- #define CFE_ES_CDS_REGISTRY_FULL ((int32)0xc4000011)
- #define CFE_ES_CDS_INVALID ((int32)0xc4000012)
- #define CFE_ES_CDS_ACCESS_ERROR ((int32)0xc4000013)
- #define CFE_ES_FILE_IO_ERR ((int32)0xc4000014)
- #define CFE_ES_RST_ACCESS_ERR ((int32)0xc4000015)
- #define CFE_ES_ERR_TASKID ((int32)0xc4000016)
- #define CFE_ES_ERR_APP_REGISTER ((int32)0xc4000017)
- #define CFE_ES_ERR_CHILD_TASK_DELETE ((int32)0xc4000018)
- #define CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK ((int32)0xc4000019)
- #define CFE_ES_CDS_BLOCK_CRC_ERR ((int32)0xc400001A)
- #define CFE_ES_MUT_SEM_DELETE_ERR ((int32)0xc400001B)
- #define CFE_ES_BIN_SEM_DELETE_ERR ((int32)0xc400001C)
- #define CFE_ES_COUNT_SEM_DELETE_ERR ((int32)0xc400001D)
- #define CFE_ES_QUEUE_DELETE_ERR ((int32)0xc400001E)
- #define CFE_ES_FILE_CLOSE_ERR ((int32)0xc400001F)
- #define CFE_ES_CDS_WRONG_TYPE_ERR ((int32)0xc4000020)
- #define CFE_ES_CDS_NOT_FOUND_ERR ((int32)0xc4000021)
- #define CFE_ES_CDS_OWNER_ACTIVE_ERR ((int32)0xc4000022)
- #define CFE_ES_APP_CLEANUP_ERR ((int32)0xc4000023)
- #define CFE_ES_TIMER_DELETE_ERR ((int32)0xc4000024)
- #define CFE_ES_BUFFER_NOT_IN_POOL ((int32)0xc4000025)
- #define CFE_ES_TASK_DELETE_ERR ((int32)0xc4000026)
- #define CFE_ES_OPERATION_TIMED_OUT ((int32)0xc4000027)
- #define CFE_ES_LIB_ALREADY_LOADED ((int32)0x44000028)
- #define CFE_ES_ERR_SYS_LOG_TRUNCATED ((int32)0x44000028)
- #define CFE_ES_NOT_IMPLEMENTED ((int32)0xc400ffff)
- #define CFE_FS_BAD_ARGUMENT ((int32)0xc6000001)
- #define CFE_FS_INVALID_PATH ((int32)0xc6000002)
- #define CFE_FS_FNAME_TOO_LONG ((int32)0xc6000003)
- #define CFE_FS_GZIP_BAD_DATA ((int32)0xc6000004)
- #define CFE_FS_GZIP_BAD_CODE_BLOCK ((int32)0xc6000005)
- #define CFE_FS_GZIP_NO_MEMORY ((int32)0xc6000006)
- #define CFE_FS_GZIP_CRC_ERROR ((int32)0xc6000007)
- #define CFE_FS_GZIP_LENGTH_ERROR ((int32)0xc6000008)

- #define CFE_FS_GZIP_WRITE_ERROR ((int32)0xc6000009)
- #define CFE_FS_GZIP_READ_ERROR ((int32)0xc600000A)
- #define CFE_FS_GZIP_OPEN_OUTPUT ((int32)0xc600000B)
- #define CFE_FS_GZIP_OPEN_INPUT ((int32)0xc600000C)
- #define CFE_FS_GZIP_READ_ERROR_HEADER ((int32)0xc600000D)
- #define CFE_FS_GZIP_INDEX_ERROR ((int32)0xc600000E)
- #define CFE_FS_GZIP_NON_ZIP_FILE ((int32)0xc600000F)
- #define CFE_FS_NOT_IMPLEMENTED ((int32)0xc600ffff)
- #define CFE_OS_ERROR (OS_ERROR)
- #define CFE_OS_INVALID_POINTER (OS_INVALID_POINTER)
- #define CFE_OS_ERROR_ADDRESS_MISALIGNED (OS_ERROR_ADDRESS_MISALIGNED)
- #define CFE_OS_ERROR_TIMEOUT (OS_ERROR_TIMEOUT)
- #define CFE_OS_INVALID_INT_NUM (OS_INVALID_INT_NUM)
- #define CFE_OS_SEM_FAILURE (OS_SEM_FAILURE)
- #define CFE_OS_SEM_TIMEOUT (OS_SEM_TIMEOUT)
- #define CFE_OS_QUEUE_EMPTY (OS_QUEUE_EMPTY)
- #define CFE_OS_QUEUE_FULL (OS_QUEUE_FULL)
- #define CFE_OS_QUEUE_TIMEOUT (OS_QUEUE_TIMEOUT)
- #define CFE_OS_QUEUE_INVALID_SIZE (OS_QUEUE_INVALID_SIZE)
- #define CFE_OS_QUEUE_ID_ERROR (OS_QUEUE_ID_ERROR)
- #define CFE_OS_ERR_NAME_TOO_LONG (OS_ERR_NAME_TOO_LONG)
- #define CFE_OS_ERR_NO_FREE_IDS (OS_ERR_NO_FREE_IDS)
- #define CFE_OS_ERR_NAME_TAKEN (OS_ERR_NAME_TAKEN)
- #define CFE_OS_ERR_INVALID_ID (OS_ERR_INVALID_ID)
- #define CFE_OS_ERR_NAME_NOT_FOUND (OS_ERR_NAME_NOT_FOUND)
- #define CFE_OS_ERR_SEM_NOT_FULL (OS_ERR_SEM_NOT_FULL)
- #define CFE_OS_ERR_INVALID_PRIORITY (OS_ERR_INVALID_PRIORITY)
- #define CFE_OS_ERROR_TASK_ID (OS_ERROR_TASK_ID)
- #define CFE_OS_SEM_UNAVAILABLE (OS_SEM_UNAVAILABLE)
- #define CFE_OS_FS_ERROR (OS_FS_ERROR)
- #define CFE_OS_FS_ERR_INVALID_POINTER (OS_FS_ERR_INVALID_POINTER)
- #define CFE_OS_FS_ERR_PATH_TOO_LONG (OS_FS_ERR_PATH_TOO_LONG)
- #define CFE_OS_FS_ERR_NAME_TOO_LONG (OS_FS_ERR_NAME_TOO_LONG)
- #define CFE_OS_FS_ERR_DRIVE_NOT_CREATED (OS_FS_ERR_DRIVE_NOT_CREATED)
- #define CFE_OSAPI_NOT_IMPLEMENTED (OS_FS_UNIMPLEMENTED)
- #define CFE_SB_TIME_OUT ((int32)0xca000001)
- #define CFE_SB_NO_MESSAGE ((int32)0xca000002)
- #define CFE_SB_BAD_ARGUMENT ((int32)0xca000003)
- #define CFE_SB_MAX_PIPES_MET ((int32)0xca000004)
- #define CFE_SB_PIPE_CR_ERR ((int32)0xca000005)
- #define CFE_SB_PIPE_RD_ERR ((int32)0xca000006)
- #define CFE_SB_MSG_TOO_BIG ((int32)0xca000007)
- #define CFE_SB_BUF_ALOC_ERR ((int32)0xca000008)
- #define CFE_SB_MAX_MSGS_MET ((int32)0xca000009)
- #define CFE_SB_MAX_DESTS_MET ((int32)0xca00000a)
- #define CFE_SB_NO_SUBSCRIBERS ((int32)0xca00000b)
- #define CFE_SB_INTERNAL_ERR ((int32)0xca00000c)
- #define CFE_SB_WRONG_MSG_TYPE ((int32)0xca00000d)
- #define CFE_SB_BUFFER_INVALID ((int32)0xca00000e)
- #define CFE_SB_NOT_IMPLEMENTED ((int32)0xca00ffff)
- #define CFE_TBL_ERR_INVALID_HANDLE ((int32)0xcc000001)

- #define CFE_TBL_ERR_INVALID_NAME ((int32)0xcc000002)
- #define CFE_TBL_ERR_INVALID_SIZE ((int32)0xcc000003)
- #define CFE_TBL_INFO_UPDATE_PENDING ((int32)0x4c000004)
- #define CFE_TBL_ERR_NEVER_LOADED ((int32)0xcc000005)
- #define CFE_TBL_ERR_REGISTRY_FULL ((int32)0xcc000006)
- #define CFE_TBL_WARN_DUPLICATE ((int32)0x4c000007)
- #define CFE_TBL_ERR_NO_ACCESS ((int32)0xcc000008)
- #define CFE_TBL_ERR_UNREGISTERED ((int32)0xcc000009)
- #define CFE_TBL_ERR_BAD_APP_ID ((int32)0xcc00000A)
- #define CFE_TBL_ERR_HANDLES_FULL ((int32)0xcc00000B)
- #define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((int32)0xcc00000C)
- #define CFE_TBL_ERR_DUPLICATE_NOT_OWNED ((int32)0xcc00000D)
- #define CFE_TBL_INFO_UPDATED ((int32)0x4c00000E)
- #define CFE_TBL_ERR_NO_BUFFER_AVAIL ((int32)0xcc00000F)
- #define CFE_TBL_ERR_DUMP_ONLY ((int32)0xcc000010)
- #define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((int32)0xcc000011)
- #define CFE_TBL_ERR_LOAD_IN_PROGRESS ((int32)0xcc000012)
- #define CFE_TBL_ERR_FILE_NOT_FOUND ((int32)0xcc000013)
- #define CFE_TBL_ERR_FILE_TOO_LARGE ((int32)0xcc000014)
- #define CFE_TBL_WARN_SHORT_FILE ((int32)0x4c000015)
- #define CFE_TBL_ERR_BAD_CONTENT_ID ((int32)0xcc000016)
- #define CFE_TBL_INFO_NO_UPDATE_PENDING ((int32)0x4c000017)
- #define CFE_TBL_INFO_TABLE_LOCKED ((int32)0x4c000018)
- #define CFE_TBL_INFO_VALIDATION_PENDING ((int32)0x4c000019)
- #define CFE_TBL_INFO_NO_VALIDATION_PENDING ((int32)0x4c00001A)
- #define CFE_TBL_ERR_BAD_SUBTYPE_ID ((int32)0xcc00001B)
- #define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((int32)0xcc00001C)
- #define CFE_TBL_ERR_NO_STD_HEADER ((int32)0xcc00001D)
- #define CFE_TBL_ERR_NO_TBL_HEADER ((int32)0xcc00001E)
- #define CFE_TBL_ERR_FILENAME_TOO_LONG ((int32)0xcc00001F)
- #define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((int32)0xcc000020)
- #define CFE_TBL_ERR_LOAD_INCOMPLETE ((int32)0xcc000021)
- #define CFE_TBL_WARN_PARTIAL_LOAD ((int32)0x4c000022)
- #define CFE_TBL_ERR_PARTIAL_LOAD ((int32)0xcc000023)
- #define CFE_TBL_INFO_DUMP_PENDING ((int32)0x4c000024)
- #define CFE_TBL_ERR_INVALID_OPTIONS ((int32)0xcc000025)
- #define CFE_TBL_WARN_NOT_CRITICAL ((int32)0x4c000026)
- #define CFE_TBL_INFO_RECOVERED_TBL ((int32)0x4c000027)
- #define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((int32)0xcc000028)
- #define CFE_TBL_ERR_BAD_PROCESSOR_ID ((int32)0xcc000029)
- #define CFE_TBL_MESSAGE_ERROR ((int32)0xcc00002a)
- #define CFE_TBL_NOT_IMPLEMENTED ((int32)0xcc00ffff)
- #define CFE_TIME_NOT_IMPLEMENTED ((int32)0xce00ffff)
- #define CFE_TIME_INTERNAL_ONLY ((int32)0xce000001)
- #define CFE_TIME_OUT_OF_RANGE ((int32)0xce000002)
- #define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((int32)0xce000003)
- #define CFE_TIME_CALLBACK_NOT_REGISTERED ((int32)0xce000004)

13.49.1 Macro Definition Documentation

13.49.1.1 CFE_ES_APP_CLEANUP_ERR

```
#define CFE_ES_APP_CLEANUP_ERR ((int32)0xc4000023)
```

Occurs when an attempt was made to Clean Up an application which involves calling Table, EVS, and SB cleanup functions, then deleting all ES resources, child tasks, and unloading the object module. The approach here is to keep going even though one of these steps had an error. There will be syslog messages detailing each problem.

Definition at line 474 of file cfe_error.h.

Referenced by CFE_ES_CleanUpApp(), CFE_ES_CleanupObjectCallback(), and CFE_ES_CleanupTaskResources().

13.49.1.2 CFE_ES_BAD_ARGUMENT

```
#define CFE_ES_BAD_ARGUMENT ((int32)0xc400000a)
```

Bad parameter passed into an ES API.

Definition at line 291 of file cfe_error.h.

Referenced by CFE_ES_CreateCDSPool(), CFE_ES_CreateChildTask(), CFE_ES_DeleteGenCounter(), CFE_ES↩_GetGenCount(), CFE_ES_GetGenCounterIDByName(), CFE_ES_IncrementGenCounter(), CFE_ES_LoadLibrary(), CFE_ES_PoolCreateEx(), CFE_ES_RebuildCDSPool(), CFE_ES_RegisterGenCounter(), CFE_ES_ResetCFE(), C↩FE_ES_SetGenCount(), and CFE_ES_SysLogSetMode().

13.49.1.3 CFE_ES_BIN_SEM_DELETE_ERR

```
#define CFE_ES_BIN_SEM_DELETE_ERR ((int32)0xc400001C)
```

Occurs when trying to delete a Binary Semaphore that belongs to a task that ES is cleaning up.

Definition at line 417 of file cfe_error.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.49.1.4 CFE_ES_BUFFER_NOT_IN_POOL

```
#define CFE_ES_BUFFER_NOT_IN_POOL ((int32)0xc4000025)
```

The specified address is not in the memory pool.

Definition at line 487 of file cfe_error.h.

Referenced by CFE_ES_GetPoolBufInfo().

13.49.1.5 CFE_ES_CDS_ACCESS_ERROR

```
#define CFE_ES_CDS_ACCESS_ERROR ((int32)0xc4000013)
```

The CDS was inaccessible

Definition at line 355 of file cfe_error.h.

Referenced by CFE_ES_GetCDSBlock(), CFE_ES_PutCDSBlock(), and CFE_ES_RebuildCDSPool().

13.49.1.6 CFE_ES_CDS_ALREADY_EXISTS

```
#define CFE_ES_CDS_ALREADY_EXISTS ((int32)0x4400000d)
```

The Application is receiving the pointer to a CDS that was already present.

Definition at line 309 of file cfe_error.h.

Referenced by CFE_ES_RegisterCDSEx().

13.49.1.7 CFE_ES_CDS_BLOCK_CRC_ERR

```
#define CFE_ES_CDS_BLOCK_CRC_ERR ((int32)0xc400001A)
```

Occurs when trying to read a CDS Data block and the CRC of the current data does not match the stored CRC for the data. Either the contents of the CDS Data Block are corrupted or the CDS Control Block is corrupted.

Definition at line 402 of file cfe_error.h.

Referenced by CFE_ES_CDSBlockRead().

13.49.1.8 CFE_ES_CDS_INSUFFICIENT_MEMORY

```
#define CFE_ES_CDS_INSUFFICIENT_MEMORY ((int32)0xc400000e)
```

The Application is requesting a CDS Block that is larger than the remaining CDS memory.

Definition at line 317 of file cfe_error.h.

13.49.1.9 CFE_ES_CDS_INVALID

```
#define CFE_ES_CDS_INVALID ((int32)0xc4000012)
```

The CDS contents are invalid.

Definition at line 348 of file cfe_error.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_RebuildCDS(), and CFE_ES_ValidateCDS().

13.49.1.10 CFE_ES_CDS_INVALID_NAME

```
#define CFE_ES_CDS_INVALID_NAME ((int32)0xc400000f)
```

The Application is requesting a CDS Block with an invalid ASCII string name. Either the name is too long (> CFE_MI↩ SSION_ES_CDS_MAX_NAME_LENGTH) or was an empty string.

Definition at line 325 of file cfe_error.h.

Referenced by CFE_ES_RegisterCDS().

13.49.1.11 CFE_ES_CDS_INVALID_SIZE

```
#define CFE_ES_CDS_INVALID_SIZE ((int32)0xc4000010)
```

The Application is requesting a CDS Block with a size of zero.

Definition at line 332 of file cfe_error.h.

Referenced by CFE_ES_RegisterCDS().

13.49.1.12 CFE_ES_CDS_NOT_FOUND_ERR

```
#define CFE_ES_CDS_NOT_FOUND_ERR ((int32)0xc4000021)
```

Occurs when a search of the Critical Data Store Registry does not find a critical data store with the specified name.

Definition at line 453 of file cfe_error.h.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_DeleteCDSCmd().

13.49.1.13 CFE_ES_CDS_OWNER_ACTIVE_ERR

```
#define CFE_ES_CDS_OWNER_ACTIVE_ERR ((int32)0xc4000022)
```

Occurs when an attempt was made to delete a CDS when an application with the same name associated with the CDS is still present. CDSs can ONLY be deleted when Applications that created them are not present in the system.

Definition at line 462 of file cfe_error.h.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_DeleteCDSCmd().

13.49.1.14 CFE_ES_CDS_REGISTRY_FULL

```
#define CFE_ES_CDS_REGISTRY_FULL ((int32)0xc4000011)
```

The CDS Registry has as many entries in it as it can hold. The CDS Registry size can be adjusted with the CFE_PL↩ ATFORM_ES_CDS_MAX_NUM_ENTRIES macro defined in the cfe_platform_cfg.h file.

Definition at line 341 of file cfe_error.h.

Referenced by CFE_ES_RegisterCDSEx().

13.49.1.15 CFE_ES_CDS_WRONG_TYPE_ERR

```
#define CFE_ES_CDS_WRONG_TYPE_ERR ((int32)0xc4000020)
```

Occurs when Table Services is trying to delete a Critical Data Store that is not a Critical Table Image or when Executive Services is trying to delete a Critical Table Image.

Definition at line 446 of file cfe_error.h.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_DeleteCDSCmd().

13.49.1.16 CFE_ES_COUNT_SEM_DELETE_ERR

```
#define CFE_ES_COUNT_SEM_DELETE_ERR ((int32)0xc400001D)
```

Occurs when trying to delete a Counting Semaphore that belongs to a task that ES is cleaning up.

Definition at line 424 of file cfe_error.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.49.1.17 CFE_ES_ERR_APP_CREATE

```
#define CFE_ES_ERR_APP_CREATE ((int32)0xc4000004)
```

There was an error loading or creating the App.

Definition at line 254 of file cfe_error.h.

Referenced by CFE_ES_AppCreate(), and CFE_ES_ParseFileEntry().

13.49.1.18 CFE_ES_ERR_APP_REGISTER

```
#define CFE_ES_ERR_APP_REGISTER ((int32)0xc4000017)
```

Occurs when the CFE_ES_RegisterApp fails.

Definition at line 381 of file cfe_error.h.

Referenced by CFE_ES_RegisterApp().

13.49.1.19 CFE_ES_ERR_APPID

```
#define CFE_ES_ERR_APPID ((int32)0xc4000001)
```

The given application ID does not reflect a currently active application.

Definition at line 236 of file cfe_error.h.

Referenced by CFE_ES_CDS_ValidateAppID(), CFE_ES_DeleteApp(), CFE_ES_GetAppIDInternal(), CFE_ES_Get↩AppInfo(), CFE_ES_GetAppName(), CFE_ES_ReloadApp(), and CFE_ES_RestartApp().

13.49.1.20 CFE_ES_ERR_APPNAME

```
#define CFE_ES_ERR_APPNAME ((int32)0xc4000002)
```

There is no match for the given application name in the current application list.

Definition at line 242 of file cfe_error.h.

Referenced by CFE_ES_DeleteCDS(), and CFE_ES_GetAppIDByName().

13.49.1.21 CFE_ES_ERR_BUFFER

```
#define CFE_ES_ERR_BUFFER ((int32)0xc4000003)
```

Invalid pointer argument (NULL)

Definition at line 248 of file cfe_error.h.

Referenced by CFE_ES_GetAppInfo(), CFE_EVS_Register(), and EVS_GetApplicationInfo().

13.49.1.22 CFE_ES_ERR_CHILD_TASK_CREATE

```
#define CFE_ES_ERR_CHILD_TASK_CREATE ((int32)0xc4000005)
```

There was an error creating a child task.

Definition at line 260 of file cfe_error.h.

Referenced by CFE_ES_CreateChildTask().

13.49.1.23 CFE_ES_ERR_CHILD_TASK_DELETE

```
#define CFE_ES_ERR_CHILD_TASK_DELETE ((int32)0xc4000018)
```

There was an error deleting a child task.

Definition at line 387 of file cfe_error.h.

Referenced by CFE_ES_CleanupObjectCallback(), and CFE_ES_DeleteChildTask().

13.49.1.24 CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK

```
#define CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK ((int32)0xc4000019)
```

There was an attempt to delete a cFE App Main Task with the CFE_ES_DeleteChildTask API.

Definition at line 394 of file cfe_error.h.

Referenced by CFE_ES_DeleteChildTask().

13.49.1.25 CFE_ES_ERR_CHILD_TASK_REGISTER

```
#define CFE_ES_ERR_CHILD_TASK_REGISTER ((int32)0xc400000b)
```

Errors occured when trying to register a child task.

Definition at line 297 of file cfe_error.h.

Referenced by CFE_ES_RegisterChildTask().

13.49.1.26 CFE_ES_ERR_LOAD_LIB

```
#define CFE_ES_ERR_LOAD_LIB ((int32)0xc4000009)
```

Could not load the shared library.

Definition at line 285 of file cfe_error.h.

Referenced by CFE_ES_LoadLibrary().

13.49.1.27 CFE_ES_ERR_MEM_BLOCK_SIZE

```
#define CFE_ES_ERR_MEM_BLOCK_SIZE ((int32)0xc4000008)
```

The block size requested is invalid.

Definition at line 279 of file cfe_error.h.

Referenced by CFE_ES_GetCDSBlock(), and CFE_ES_GetPoolBuf().

13.49.1.28 CFE_ES_ERR_MEM_HANDLE

```
#define CFE_ES_ERR_MEM_HANDLE ((int32)0xc4000007)
```

The Memory Pool handle is invalid.

Definition at line 273 of file cfe_error.h.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetMemPoolStats(), CFE_ES_Get↩PoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_PutCDSBlock(), and CFE_ES_PutPoolBuf().

13.49.1.29 CFE_ES_ERR_SHELL_CMD

```
#define CFE_ES_ERR_SHELL_CMD ((int32)0xc400000c)
```

Error occured ehen trying to pass a system call to the OS shell

Definition at line 303 of file cfe_error.h.

Referenced by CFE_ES_ShellOutputCommand().

13.49.1.30 CFE_ES_ERR_SYS_LOG_FULL

```
#define CFE_ES_ERR_SYS_LOG_FULL ((int32)0xc4000006)
```

The cFE system Log is full. This error means the message was not logged at all

Definition at line 267 of file cfe_error.h.

Referenced by CFE_ES_SysLogAppend_Unsync().

13.49.1.31 CFE_ES_ERR_SYS_LOG_TRUNCATED

```
#define CFE_ES_ERR_SYS_LOG_TRUNCATED ((int32)0x44000028)
```

cFE system Log message truncated. This information code means the last syslog message was truncated due to insufficient space in the log buffer.

Definition at line 517 of file cfe_error.h.

Referenced by CFE_ES_SysLogAppend_Unsync().

13.49.1.32 CFE_ES_ERR_TASKID

```
#define CFE_ES_ERR_TASKID ((int32)0xc4000016)
```

Occurs when the Task ID passed into CFE_ES_GetTaskInfo is invalid.

Definition at line 375 of file cfe_error.h.

Referenced by CFE_ES_DeleteChildTask(), and CFE_ES_GetTaskInfo().

13.49.1.33 CFE_ES_FILE_CLOSE_ERR

```
#define CFE_ES_FILE_CLOSE_ERR ((int32)0xc400001F)
```

Occurs when trying to close a file that belongs to a task that ES is cleaning up.

Definition at line 438 of file cfe_error.h.

13.49.1.34 CFE_ES_FILE_IO_ERR

```
#define CFE_ES_FILE_IO_ERR ((int32)0xc4000014)
```

Occurs when a file operation fails

Definition at line 362 of file cfe_error.h.

Referenced by CFE_ES_ERLogDump(), CFE_ES_ReloadApp(), and CFE_ES_SysLogDump().

13.49.1.35 CFE_ES_LIB_ALREADY_LOADED

```
#define CFE_ES_LIB_ALREADY_LOADED ((int32)0x44000028)
```

Occurs if CFE_ES_LoadLibrary() detects that the requested library name is already loaded.

Definition at line 508 of file cfe_error.h.

Referenced by CFE_ES_LoadLibrary().

13.49.1.36 CFE_ES_MUT_SEM_DELETE_ERR

```
#define CFE_ES_MUT_SEM_DELETE_ERR ((int32)0xc400001B)
```

Occurs when trying to delete a Mutex that belongs to a task that ES is cleaning up.

Definition at line 409 of file cfe_error.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.49.1.37 CFE_ES_NOT_IMPLEMENTED

```
#define CFE_ES_NOT_IMPLEMENTED ((int32)0xc400ffff)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 526 of file cfe_error.h.

Referenced by CFE_ES_RegisterCDS(), and CFE_ES_ResetCFE().

13.49.1.38 CFE_ES_OPERATION_TIMED_OUT

```
#define CFE_ES_OPERATION_TIMED_OUT ((int32)0xc4000027)
```

Occurs if the timeout for a given operation was exceeded

Definition at line 501 of file cfe_error.h.

Referenced by CFE_ES_MainTaskSyncDelay(), and CFE_ES_WaitForSystemState().

13.49.1.39 CFE_ES_QUEUE_DELETE_ERR

```
#define CFE_ES_QUEUE_DELETE_ERR ((int32)0xc400001E)
```

Occurs when trying to delete a Queue that belongs to a task that ES is cleaning up.

Definition at line 431 of file cfe_error.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.49.1.40 CFE_ES_RST_ACCESS_ERR

```
#define CFE_ES_RST_ACCESS_ERR ((int32)0xc4000015)
```

Occurs when the BSP is not successful in returning the reset area address.

Definition at line 369 of file cfe_error.h.

Referenced by CFE_ES_ERLogDump().

13.49.1.41 CFE_ES_TASK_DELETE_ERR

```
#define CFE_ES_TASK_DELETE_ERR ((int32)0xc4000026)
```

Occurs when trying to delete a task that ES is cleaning up.

Definition at line 495 of file cfe_error.h.

Referenced by CFE_ES_CleanupTaskResources().

13.49.1.42 CFE_ES_TIMER_DELETE_ERR

```
#define CFE_ES_TIMER_DELETE_ERR ((int32)0xc4000024)
```

Occurs when trying to delete a Timer that belongs to a task that ES is cleaning up.

Definition at line 481 of file cfe_error.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.49.1.43 CFE_EVENTS_SERVICE

```
#define CFE_EVENTS_SERVICE ((int32)0x02000000)
```

Definition at line 99 of file cfe_error.h.

13.49.1.44 CFE_EVS_APP_FILTER_OVERLOAD

```
#define CFE_EVS_APP_FILTER_OVERLOAD ((int32)0xc2000004)
```

Number of Application event filters input upon registration is greater than CFE_PLATFORM_EVS_MAX_EVENT_FIL↩TERS

Definition at line 183 of file cfe_error.h.

Referenced by CFE_EVS_AddEventFilterCmd().

13.49.1.45 CFE_EVS_APP_ILLEGAL_APP_ID

```
#define CFE_EVS_APP_ILLEGAL_APP_ID ((int32)0xc2000003)
```

Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS

Definition at line 176 of file cfe_error.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_CleanUpApp(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SetFilterCmd(), EVS_GetAppID(), and EVS_GetApplicationInfo().

13.49.1.46 CFE_EVS_APP_NOT_REGISTERED

```
#define CFE_EVS_APP_NOT_REGISTERED ((int32)0xc2000002)
```

Calling application never previously called CFE_EVS_Register()

Definition at line 169 of file cfe_error.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), EVS_GetApplicationInfo(), and EVS_NotRegistered().

13.49.1.47 CFE_EVS_EVT_NOT_REGISTERED

```
#define CFE_EVS_EVT_NOT_REGISTERED ((int32)0xc2000006)
```

CFE_EVS_ResetFilter() EventID argument was not found in any event filter registered by the calling application.

Definition at line 198 of file cfe_error.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

13.49.1.48 CFE_EVS_FILE_WRITE_ERROR

```
#define CFE_EVS_FILE_WRITE_ERROR ((int32)0xc2000007)
```

A file write error occurred while processing an EVS command

Definition at line 204 of file cfe_error.h.

Referenced by CFE_EVS_WriteAppDataFileCmd(), and CFE_EVS_WriteLogDataFileCmd().

13.49.1.49 CFE_EVS_FUNCTION_DISABLED

```
#define CFE_EVS_FUNCTION_DISABLED ((int32)0xc2000009)
```

EVS command sent that requires a feature currently turned off This is to differentiate between "NOT_IMPLEMENTED" where the feature IS implemented but it is disabled at runtime.

Definition at line 217 of file cfe_error.h.

Referenced by CFE_EVS_ClearLogCmd(), CFE_EVS_SetLogModeCmd(), and CFE_EVS_WriteLogDataFileCmd().

13.49.1.50 CFE_EVS_INVALID_PARAMETER

```
#define CFE_EVS_INVALID_PARAMETER ((int32)0xc2000008)
```

Invalid parameter supplied to EVS command

Definition at line 210 of file cfe_error.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_Disable↩PortsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_EnableEventTypeCmd(), CFE_EVS_EnablePorts↩Cmd(), CFE_EVS_SetEventFormatModeCmd(), and CFE_EVS_SetLogModeCmd().

13.49.1.51 CFE_EVS_NOT_IMPLEMENTED

```
#define CFE_EVS_NOT_IMPLEMENTED ((int32)0xc200ffff)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 226 of file cfe_error.h.

13.49.1.52 CFE_EVS_RESET_AREA_POINTER

```
#define CFE_EVS_RESET_AREA_POINTER ((int32)0xc2000005)
```

Could not get pointer to the ES Reset area, so we could not get the pointer to the EVS Log.

Definition at line 190 of file cfe_error.h.

Referenced by CFE_EVS_EarlyInit().

13.49.1.53 CFE_EVS_UNKNOWN_FILTER

```
#define CFE_EVS_UNKNOWN_FILTER ((int32)0xc2000001)
```

CFE_EVS_Register() FilterScheme parameter was illegal

Definition at line 163 of file cfe_error.h.

Referenced by CFE_EVS_Register().

13.49.1.54 CFE_EXECUTIVE_SERVICE

```
#define CFE_EXECUTIVE_SERVICE ((int32)0x04000000)
```

Definition at line 100 of file cfe_error.h.

13.49.1.55 CFE_FILE_SERVICE

```
#define CFE_FILE_SERVICE ((int32)0x06000000)
```

Definition at line 101 of file cfe_error.h.

13.49.1.56 CFE_FS_BAD_ARGUMENT

```
#define CFE_FS_BAD_ARGUMENT ((int32)0xc6000001)
```

Definition at line 538 of file cfe_error.h.

13.49.1.57 CFE_FS_FNAME_TOO_LONG

```
#define CFE_FS_FNAME_TOO_LONG ((int32)0xc6000003)
```

Definition at line 550 of file cfe_error.h.

13.49.1.58 CFE_FS_GZIP_BAD_CODE_BLOCK

```
#define CFE_FS_GZIP_BAD_CODE_BLOCK ((int32)0xc6000005)
```

Definition at line 561 of file cfe_error.h.

13.49.1.59 CFE_FS_GZIP_BAD_DATA

```
#define CFE_FS_GZIP_BAD_DATA ((int32)0xc6000004)
```

Definition at line 555 of file cfe_error.h.

13.49.1.60 CFE_FS_GZIP_CRC_ERROR

```
#define CFE_FS_GZIP_CRC_ERROR ((int32)0xc6000007)
```

Definition at line 573 of file cfe_error.h.

13.49.1.61 CFE_FS_GZIP_INDEX_ERROR

```
#define CFE_FS_GZIP_INDEX_ERROR ((int32)0xc600000E)
```

Definition at line 619 of file cfe_error.h.

13.49.1.62 CFE_FS_GZIP_LENGTH_ERROR

```
#define CFE_FS_GZIP_LENGTH_ERROR ((int32)0xc6000008)
```

Definition at line 579 of file cfe_error.h.

13.49.1.63 CFE_FS_GZIP_NO_MEMORY

```
#define CFE_FS_GZIP_NO_MEMORY ((int32)0xc6000006)
```

Definition at line 567 of file cfe_error.h.

13.49.1.64 CFE_FS_GZIP_NON_ZIP_FILE

```
#define CFE_FS_GZIP_NON_ZIP_FILE ((int32)0xc600000F)
```

Definition at line 624 of file cfe_error.h.

13.49.1.65 CFE_FS_GZIP_OPEN_INPUT

```
#define CFE_FS_GZIP_OPEN_INPUT ((int32)0xc600000C)
```

Definition at line 606 of file cfe_error.h.

13.49.1.66 CFE_FS_GZIP_OPEN_OUTPUT

```
#define CFE_FS_GZIP_OPEN_OUTPUT ((int32)0xc600000B)
```

Definition at line 598 of file cfe_error.h.

13.49.1.67 CFE_FS_GZIP_READ_ERROR

```
#define CFE_FS_GZIP_READ_ERROR ((int32)0xc600000A)
```

Definition at line 590 of file cfe_error.h.

13.49.1.68 CFE_FS_GZIP_READ_ERROR_HEADER

```
#define CFE_FS_GZIP_READ_ERROR_HEADER ((int32)0xc600000D)
```

Definition at line 613 of file cfe_error.h.

13.49.1.69 CFE_FS_GZIP_WRITE_ERROR

```
#define CFE_FS_GZIP_WRITE_ERROR ((int32)0xc6000009)
```

Definition at line 585 of file cfe_error.h.

13.49.1.70 CFE_FS_INVALID_PATH

```
#define CFE_FS_INVALID_PATH ((int32)0xc6000002)
```

Definition at line 544 of file cfe_error.h.

13.49.1.71 CFE_FS_NOT_IMPLEMENTED

```
#define CFE_FS_NOT_IMPLEMENTED ((int32)0xc600ffff)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 633 of file cfe_error.h.

13.49.1.72 CFE_GENERIC_SERVICE

```
#define CFE_GENERIC_SERVICE ((int32)0x08000000)
```

Definition at line 102 of file cfe_error.h.

13.49.1.73 CFE_OS_ERR_INVALID_ID

```
#define CFE_OS_ERR_INVALID_ID (OS_ERR_INVALID_ID)
```

Definition at line 733 of file cfe_error.h.

13.49.1.74 CFE_OS_ERR_INVALID_PRIORITY

```
#define CFE_OS_ERR_INVALID_PRIORITY (OS_ERR_INVALID_PRIORITY)
```

Definition at line 751 of file cfe_error.h.

13.49.1.75 CFE_OS_ERR_NAME_NOT_FOUND

```
#define CFE_OS_ERR_NAME_NOT_FOUND (OS_ERR_NAME_NOT_FOUND)
```

Definition at line 739 of file cfe_error.h.

13.49.1.76 CFE_OS_ERR_NAME_TAKEN

```
#define CFE_OS_ERR_NAME_TAKEN (OS_ERR_NAME_TAKEN)
```

Definition at line 727 of file cfe_error.h.

13.49.1.77 CFE_OS_ERR_NAME_TOO_LONG

```
#define CFE_OS_ERR_NAME_TOO_LONG (OS_ERR_NAME_TOO_LONG)
```

Definition at line 715 of file cfe_error.h.

13.49.1.78 CFE_OS_ERR_NO_FREE_IDS

```
#define CFE_OS_ERR_NO_FREE_IDS (OS_ERR_NO_FREE_IDS)
```

Definition at line 721 of file cfe_error.h.

13.49.1.79 CFE_OS_ERR_SEM_NOT_FULL

```
#define CFE_OS_ERR_SEM_NOT_FULL (OS_ERR_SEM_NOT_FULL)
```

Definition at line 745 of file cfe_error.h.

13.49.1.80 CFE_OS_ERROR

```
#define CFE_OS_ERROR (OS_ERROR)
```

Definition at line 643 of file cfe_error.h.

13.49.1.81 CFE_OS_ERROR_ADDRESS_MISALIGNED

```
#define CFE_OS_ERROR_ADDRESS_MISALIGNED (OS_ERROR_ADDRESS_MISALIGNED)
```

Definition at line 655 of file cfe_error.h.

13.49.1.82 CFE_OS_ERROR_TASK_ID

```
#define CFE_OS_ERROR_TASK_ID (OS_ERROR_TASK_ID)
```

Definition at line 757 of file cfe_error.h.

13.49.1.83 CFE_OS_ERROR_TIMEOUT

```
#define CFE_OS_ERROR_TIMEOUT (OS_ERROR_TIMEOUT)
```

Definition at line 661 of file cfe_error.h.

13.49.1.84 CFE_OS_FS_ERR_DRIVE_NOT_CREATED

```
#define CFE_OS_FS_ERR_DRIVE_NOT_CREATED (OS_FS_ERR_DRIVE_NOT_CREATED)
```

Definition at line 793 of file cfe_error.h.

13.49.1.85 CFE_OS_FS_ERR_INVALID_POINTER

```
#define CFE_OS_FS_ERR_INVALID_POINTER (OS_FS_ERR_INVALID_POINTER)
```

Definition at line 775 of file cfe_error.h.

13.49.1.86 CFE_OS_FS_ERR_NAME_TOO_LONG

```
#define CFE_OS_FS_ERR_NAME_TOO_LONG (OS_FS_ERR_NAME_TOO_LONG)
```

Definition at line 787 of file cfe_error.h.

13.49.1.87 CFE_OS_FS_ERR_PATH_TOO_LONG

```
#define CFE_OS_FS_ERR_PATH_TOO_LONG (OS_FS_ERR_PATH_TOO_LONG)
```

Definition at line 781 of file cfe_error.h.

13.49.1.88 CFE_OS_FS_ERROR

```
#define CFE_OS_FS_ERROR (OS_FS_ERROR)
```

Definition at line 769 of file cfe_error.h.

13.49.1.89 CFE_OS_INVALID_INT_NUM

```
#define CFE_OS_INVALID_INT_NUM (OS_INVALID_INT_NUM)
```

Definition at line 667 of file cfe_error.h.

13.49.1.90 CFE_OS_INVALID_POINTER

```
#define CFE_OS_INVALID_POINTER (OS_INVALID_POINTER)
```

Definition at line 649 of file cfe_error.h.

13.49.1.91 CFE_OS_QUEUE_EMPTY

```
#define CFE_OS_QUEUE_EMPTY (OS_QUEUE_EMPTY)
```

Definition at line 685 of file cfe_error.h.

13.49.1.92 CFE_OS_QUEUE_FULL

```
#define CFE_OS_QUEUE_FULL (OS_QUEUE_FULL)
```

Definition at line 691 of file cfe_error.h.

13.49.1.93 CFE_OS_QUEUE_ID_ERROR

```
#define CFE_OS_QUEUE_ID_ERROR (OS_QUEUE_ID_ERROR)
```

Definition at line 709 of file cfe_error.h.

13.49.1.94 CFE_OS_QUEUE_INVALID_SIZE

```
#define CFE_OS_QUEUE_INVALID_SIZE (OS_QUEUE_INVALID_SIZE)
```

Definition at line 703 of file cfe_error.h.

13.49.1.95 CFE_OS_QUEUE_TIMEOUT

```
#define CFE_OS_QUEUE_TIMEOUT (OS_QUEUE_TIMEOUT)
```

Definition at line 697 of file cfe_error.h.

13.49.1.96 CFE_OS_SEM_FAILURE

```
#define CFE_OS_SEM_FAILURE (OS_SEM_FAILURE)
```

Definition at line 673 of file cfe_error.h.

13.49.1.97 CFE_OS_SEM_TIMEOUT

```
#define CFE_OS_SEM_TIMEOUT (OS_SEM_TIMEOUT)
```

Definition at line 679 of file cfe_error.h.

13.49.1.98 CFE_OS_SEM_UNAVAILABLE

```
#define CFE_OS_SEM_UNAVAILABLE (OS_SEM_UNAVAILABLE)
```

Definition at line 763 of file cfe_error.h.

13.49.1.99 CFE_OSAPI_NOT_IMPLEMENTED

```
#define CFE_OSAPI_NOT_IMPLEMENTED (OS_FS_UNIMPLEMENTED)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 802 of file cfe_error.h.

13.49.1.100 CFE_SB_BAD_ARGUMENT

```
#define CFE_SB_BAD_ARGUMENT ((int32)0xca000003)
```

A parameter given by a caller to a Software Bus API did not pass validation checks.

Definition at line 829 of file cfe_error.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_MessageStringGet(), CFE_SB_PutDestinationBlk(), CFE_SB_RcvMsg(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

13.49.1.101 CFE_SB_BUF_ALOC_ERR

```
#define CFE_SB_BUF_ALOC_ERR ((int32)0xca000008)
```

This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BU↩F_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Definition at line 883 of file cfe_error.h.

Referenced by CFE_SB_SendMsgFull(), and CFE_SB_SubscribeFull().

13.49.1.102 CFE_SB_BUFFER_INVALID

```
#define CFE_SB_BUFFER_INVALID ((int32)0xca00000e)
```

This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Definition at line 937 of file cfe_error.h.

Referenced by CFE_SB_ZeroCopyReleaseDesc().

13.49.1.103 CFE_SB_INTERNAL_ERR

```
#define CFE_SB_INTERNAL_ERR ((int32)0xca00000c)
```

This error code will be returned by the CFE_SB_Subscribe API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Definition at line 920 of file cfe_error.h.

13.49.1.104 CFE_SB_MAX_DESTS_MET

```
#define CFE_SB_MAX_DESTS_MET ((int32)0xca00000a)
```

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER_PKT.

Definition at line 903 of file cfe_error.h.

Referenced by CFE_SB_SubscribeFull().

13.49.1.105 CFE_SB_MAX_MSGS_MET

```
#define CFE_SB_MAX_MSGS_MET ((int32)0xca000009)
```

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.

Definition at line 892 of file cfe_error.h.

Referenced by CFE_SB_SubscribeFull().

13.49.1.106 CFE_SB_MAX_PIPES_MET

```
#define CFE_SB_MAX_PIPES_MET ((int32)0xca000004)
```

This error code will be returned from CFE_SB_CreatePipe when the SB cannot accomodate the request to create a pipe because the maximum number of pipes (CFE_PLATFORM_SB_MAX_PIPES) are in use. This configuration parameter is defined in the cfe_platform_cfg.h file.

Definition at line 839 of file cfe_error.h.

Referenced by CFE_SB_CreatePipe().

13.49.1.107 CFE_SB_MSG_TOO_BIG

```
#define CFE_SB_MSG_TOO_BIG ((int32)0xca000007)
```

The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h

Definition at line 870 of file cfe_error.h.

Referenced by CFE_SB_SendMsgFull().

13.49.1.108 CFE_SB_NO_MESSAGE

```
#define CFE_SB_NO_MESSAGE ((int32)0xca000002)
```

When "Polling" a pipe for a message in CFE_SB_RcvMsg, this return value indicates that there was not a message on the pipe.

Definition at line 821 of file cfe_error.h.

Referenced by CFE_SB_ReadQueue().

13.49.1.109 CFE_SB_NO_SUBSCRIBERS

```
#define CFE_SB_NO_SUBSCRIBERS ((int32)0xca00000b)
```

This error code is returned by the CFE_SB_Unsubscribe API if there has not been an entry in the routing tables for the MsgId/PipeId given as parameters.

Definition at line 911 of file cfe_error.h.

13.49.1.110 CFE_SB_NOT_IMPLEMENTED

```
#define CFE_SB_NOT_IMPLEMENTED ((int32)0xca00ffff)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 947 of file cfe_error.h.

13.49.1.111 CFE_SB_PIPE_CR_ERR

```
#define CFE_SB_PIPE_CR_ERR ((int32)0xca000005)
```

The maximum number of queues(OS_MAX_QUEUES) are in use. Or possibly a lower level problem with creating the underlying queue has occurred such as a lack of memory. If the latter is the problem, the status code displayed in the event must be tracked.

Definition at line 849 of file cfe_error.h.

Referenced by CFE_SB_CreatePipe().

13.49.1.112 CFE_SB_PIPE_RD_ERR

```
#define CFE_SB_PIPE_RD_ERR ((int32)0xca000006)
```

This return value indicates an error at the Queue read level. This error typically cannot be corrected by the caller. Some possible causes are: queue was not properly initialized or created, the number of bytes read from the queue was not the number of bytes requested in the read. The queue id is invalid. Similar errors regarding the pipe will be caught by higher level code in the Software Bus.

Definition at line 861 of file cfe_error.h.

Referenced by CFE_SB_ReadQueue().

13.49.1.113 CFE_SB_TIME_OUT

```
#define CFE_SB_TIME_OUT ((int32)0xca000001)
```

In CFE_SB_RcvMsg, this return value indicates that a packet has not been received in the time given in the "timeout" parameter.

Definition at line 813 of file cfe_error.h.

Referenced by CFE_ES_TaskMain(), and CFE_SB_ReadQueue().

13.49.1.114 CFE_SB_WRONG_MSG_TYPE

```
#define CFE_SB_WRONG_MSG_TYPE ((int32)0xca00000d)
```

This error code will be returned when a request such as ...SetMsgTime is made on a packet that does not include a field for msg time.

Definition at line 928 of file cfe_error.h.

Referenced by CFE_SB_SetCmdCode(), and CFE_SB_SetMsgTime().

13.49.1.115 CFE_SERVICE_BITMASK

```
#define CFE_SERVICE_BITMASK ((int32)0x0e000000)
```

Definition at line 97 of file cfe_error.h.

13.49.1.116 CFE_SEVERITY_BITMASK

```
#define CFE_SEVERITY_BITMASK ((int32)0xc0000000)
```

Definition at line 88 of file cfe_error.h.

13.49.1.117 CFE_SEVERITY_ERROR

```
#define CFE_SEVERITY_ERROR ((int32)0xc0000000)
```

Definition at line 92 of file cfe_error.h.

13.49.1.118 CFE_SEVERITY_INFO

```
#define CFE_SEVERITY_INFO ((int32)0x40000000)
```

Definition at line 91 of file cfe_error.h.

13.49.1.119 CFE_SEVERITY_SUCCESS

```
#define CFE_SEVERITY_SUCCESS ((int32)0x00000000)
```

Definition at line 90 of file cfe_error.h.

13.49.1.120 CFE_SOFTWARE_BUS_SERVICE

```
#define CFE_SOFTWARE_BUS_SERVICE ((int32)0x0a000000)
```

Definition at line 103 of file cfe_error.h.

13.49.1.121 CFE_STATUS_BAD_COMMAND_CODE

```
#define CFE_STATUS_BAD_COMMAND_CODE ((int32)0xc8000004)
```

This error code will be returned when a message identification process determined that the command code is does not correspond to any known value

Definition at line 141 of file cfe_error.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.49.1.122 CFE_STATUS_NO_COUNTER_INCREMENT

```
#define CFE_STATUS_NO_COUNTER_INCREMENT ((int32)0x48000001)
```

Informational code indicating that a command was processed successfully but that the command counter should *not* be incremented.

Definition at line 120 of file cfe_error.h.

Referenced by CFE_EVS_ReportHousekeepingCmd(), and CFE_EVS_ResetCountersCmd().

13.49.1.123 CFE_STATUS_NOT_IMPLEMENTED

```
#define CFE_STATUS_NOT_IMPLEMENTED ((int32)0xc800ffff)
```

Current version does not have the function or the feature of the function implemented. This could be due to either an early build for this platform or the platform does not support the specified feature.

Definition at line 150 of file cfe_error.h.

13.49.1.124 CFE_STATUS_UNKNOWN_MSG_ID

```
#define CFE_STATUS_UNKNOWN_MSG_ID ((int32)0xc8000003)
```

This error code will be returned when a message identification process determined that the message ID does not correspond to a known value

Definition at line 134 of file cfe_error.h.

13.49.1.125 CFE_STATUS_WRONG_MSG_LENGTH

```
#define CFE_STATUS_WRONG_MSG_LENGTH ((int32)0xc8000002)
```

Definition at line 127 of file cfe_error.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.49.1.126 CFE_SUCCESS

```
#define CFE_SUCCESS (0)
```

Operation was performed successfully

Definition at line 114 of file cfe_error.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CDS_EarlyInit(), CFE_ES_CDS_ValidateAppID(), CFE_ES_CDS↩ BlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CleanUpApp(), CFE_ES_CleanupObjectCallback(), CFE_ES↩ _CleanupTaskResources(), CFE_ES_ClearERLogCmd(), CFE_ES_ClearSyslogCmd(), CFE_ES_CreateCDSPool(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteApp(), CFE_ES_DeleteCDS(), CFE_ES_↩ DeleteCDSCmd(), CFE_ES_DeleteChildTask(), CFE_ES_DeleteGenCounter(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_ExitApp(), CFE_ES_ExitChildTask(), CFE_ES_GetAppIDByName(), CFE_ES↩ _GetAppIDInternal(), CFE_ES_GetAppInfo(), CFE_ES_GetAppName(), CFE_ES_GetGenCount(), CFE_ES_Get↩ GenCounterIDByName(), CFE_ES_GetMemPoolStats(), CFE_ES_GetTaskInfo(), CFE_ES_HousekeepingCmd(), C↩ FE_ES_IncrementGenCounter(), CFE_ES_InitCDSRegistry(), CFE_ES_InitializeCDS(), CFE_ES_ListApplications(), CFE_ES_ListResources(), CFE_ES_ListResourcesDebug(), CFE_ES_ListTasks(), CFE_ES_LoadLibrary(), CFE_↩ ES_LockCDSRegistry(), CFE_ES_Main(), CFE_ES_MainTaskSyncDelay(), CFE_ES_NoopCmd(), CFE_ES_Over↩ WriteSyslogCmd(), CFE_ES_PoolCreateEx(), CFE_ES_ProcessControlRequest(), CFE_ES_ProcessCoreException(),

CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_RegisterApp(), CFE_ES_RegisterCDS(), CFE_ES_RegisterCDSEx(), CFE_ES_RegisterChildTask(), CFE_ES_RegisterGenCounter(), CFE_ES_ReloadApp(), CFE_ES_ReloadAppCmd(), CFE_ES_ResetCountersCmd(), CFE_ES_ResetPRCountCmd(), CFE_ES_RestartApp(), CFE_ES_RestartAppCmd(), CFE_ES_RestartCmd(), CFE_ES_RunLoop(), CFE_ES_SendMemPoolStatsCmd(), CFE_ES_SetGenCount(), CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_ShellCmd(), CFE_ES_ShellOutputCommand(), CFE_ES_StartAppCmd(), CFE_ES_StartPerfDataCmd(), CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_SysLogAppend_Unsync(), CFE_ES_SysLogDump(), CFE_ES_SysLogSetMode(), CFE_ES_TaskInit(), CFE_ES_TaskMain(), CFE_ES_UnlockCDSRegistry(), CFE_ES_ValidateCDS(), CFE_ES_WaitForSystemState(), CFE_ES_WriteERLogCmd(), CFE_ES_WriteSyslogCmd(), CFE_ES_WriteToERLog(), CFE_EVS_AddEventFilterCmd(), CFE_EVS_CleanUpApp(), CFE_EVS_ClearLogCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_EarlyInit(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_EnableEventTypeCmd(), CFE_EVS_EnablePortsCmd(), CFE_EVS_NoopCmd(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_Register(), CFE_EVS_ResetAllFilters(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SendEvent(), CFE_EVS_SendEventWithAppID(), CFE_EVS_SendTimedEvent(), CFE_EVS_SetEventFormatModeCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_SetLogModeCmd(), CFE_EVS_TaskInit(), CFE_EVS_TaskMain(), CFE_EVS_Unregister(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_AddDest(), CFE_SB_AppInit(), CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DecrBufUseCnt(), CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_DisableSubReportingCmd(), CFE_SB_EarlyInit(), CFE_SB_EnableRouteCmd(), CFE_SB_EnableSubReportingCmd(), CFE_SB_GetAppTskName(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_IncrCmdCtr(), CFE_SB_InitBuffers(), CFE_SB_NoopCmd(), CFE_SB_PutDestinationBlk(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_RemoveDest(), CFE_SB_ResetCountersCmd(), CFE_SB_ReturnBufferToPool(), CFE_SB_SendHKTlmCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendMapInfoCmd(), CFE_SB_SendMsgFull(), CFE_SB_SendPipeInfo(), CFE_SB_SendPipeInfoCmd(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRoutingInfoCmd(), CFE_SB_SendRtgInfo(), CFE_SB_SendStatsCmd(), CFE_SB_SetCmdCode(), CFE_SB_SetMsgTime(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_TaskMain(), CFE_SB_UnsubscribeFull(), CFE_SB_ValidateMsgId(), CFE_SB_ValidatePipeId(), CFE_SB_ZeroCopyPass(), CFE_SB_ZeroCopyReleaseAppId(), CFE_SB_ZeroCopyReleaseDesc(), CFE_SB_ZeroCopyReleasePtr(), CFE_SB_ZeroCopySend(), EVS_GetAppID(), EVS_GetApplicationInfo(), and EVS_SendEvent().

13.49.1.127 CFE_TABLE_SERVICE

```
#define CFE_TABLE_SERVICE ((int32)0x0c000000)
```

Definition at line 104 of file cfe_error.h.

13.49.1.128 CFE_TBL_ERR_BAD_APP_ID

```
#define CFE_TBL_ERR_BAD_APP_ID ((int32)0xcc00000A)
```

The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.

Definition at line 1030 of file cfe_error.h.

13.49.1.129 CFE_TBL_ERR_BAD_CONTENT_ID

```
#define CFE_TBL_ERR_BAD_CONTENT_ID ((int32)0xcc000016)
```

The calling Application called CFE_TBL_Load with a filename that specified a file whose content ID was not that of a table image.

Definition at line 1120 of file cfe_error.h.

13.49.1.130 CFE_TBL_ERR_BAD_PROCESSOR_ID

```
#define CFE_TBL_ERR_BAD_PROCESSOR_ID ((int32)0xcc000029)
```

The selected table file failed validation for Processor ID. The platform configuration file has verification of table files enabled for Processor ID and an attempt was made to load a table with an invalid Processor ID in the table file header.

Definition at line 1281 of file cfe_error.h.

13.49.1.131 CFE_TBL_ERR_BAD_SPACECRAFT_ID

```
#define CFE_TBL_ERR_BAD_SPACECRAFT_ID ((int32)0xcc000028)
```

The selected table file failed validation for Spacecraft ID. The platform configuration file has verification of table files enabled for Spacecraft ID and an attempt was made to load a table with an invalid Spacecraft ID in the table file header.

Definition at line 1271 of file cfe_error.h.

13.49.1.132 CFE_TBL_ERR_BAD_SUBTYPE_ID

```
#define CFE_TBL_ERR_BAD_SUBTYPE_ID ((int32)0xcc00001B)
```

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.

Definition at line 1151 of file cfe_error.h.

13.49.1.133 CFE_TBL_ERR_DUMP_ONLY

```
#define CFE_TBL_ERR_DUMP_ONLY ((int32)0xcc000010)
```

The calling Application has attempted to perform a load on a table that was created with "Dump Only" attributes.

Definition at line 1076 of file cfe_error.h.

13.49.1.134 CFE_TBL_ERR_DUPLICATE_DIFF_SIZE

```
#define CFE_TBL_ERR_DUPLICATE_DIFF_SIZE ((int32)0xcc00000C)
```

An application attempted to register a table with the same name as a table that is already in the registry. The size of the new table is different from the size already in the registry.

Definition at line 1045 of file cfe_error.h.

13.49.1.135 CFE_TBL_ERR_DUPLICATE_NOT_OWNED

```
#define CFE_TBL_ERR_DUPLICATE_NOT_OWNED ((int32)0xcc00000D)
```

An application attempted to register a table with the same name as a table that is already in the registry. The previously registered table is owned by a different application.

Definition at line 1053 of file cfe_error.h.

13.49.1.136 CFE_TBL_ERR_FILE_FOR_WRONG_TABLE

```
#define CFE_TBL_ERR_FILE_FOR_WRONG_TABLE ((int32)0xcc000020)
```

The calling Application tried to load a table using a file whose header indicated that it was for a different table.

Definition at line 1187 of file cfe_error.h.

13.49.1.137 CFE_TBL_ERR_FILE_NOT_FOUND

```
#define CFE_TBL_ERR_FILE_NOT_FOUND ((int32)0xcc000013)
```

The calling Application called CFE_TBL_Load with a bad filename.

Definition at line 1096 of file cfe_error.h.

13.49.1.138 CFE_TBL_ERR_FILE_SIZE_INCONSISTENT

```
#define CFE_TBL_ERR_FILE_SIZE_INCONSISTENT ((int32)0xcc00001C)
```

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.

Definition at line 1158 of file cfe_error.h.

13.49.1.139 CFE_TBL_ERR_FILE_TOO_LARGE

```
#define CFE_TBL_ERR_FILE_TOO_LARGE ((int32)0xcc000014)
```

The calling Application called CFE_TBL_Load with a filename that specified a file that contained more data than the size of the table OR which contained more data than specified in the table header.

Definition at line 1104 of file cfe_error.h.

13.49.1.140 CFE_TBL_ERR_FILENAME_TOO_LONG

```
#define CFE_TBL_ERR_FILENAME_TOO_LONG ((int32)0xcc00001F)
```

The calling Application tried to load a table using a filename that was too long.

Definition at line 1179 of file cfe_error.h.

13.49.1.141 CFE_TBL_ERR_HANDLES_FULL

```
#define CFE_TBL_ERR_HANDLES_FULL ((int32)0xcc00000B)
```

An application attempted to create a table and the Table Handle Array already used all CFE_PLATFORM_TBL_MAX↩ _NUM_HANDLES in it.

Definition at line 1037 of file cfe_error.h.

13.49.1.142 CFE_TBL_ERR_ILLEGAL_SRC_TYPE

```
#define CFE_TBL_ERR_ILLEGAL_SRC_TYPE ((int32)0xcc000011)
```

The calling Application called CFE_TBL_Load with an illegal value for the second parameter.

Definition at line 1083 of file cfe_error.h.

13.49.1.143 CFE_TBL_ERR_INVALID_HANDLE

```
#define CFE_TBL_ERR_INVALID_HANDLE ((int32)0xcc000001)
```

The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Definition at line 959 of file cfe_error.h.

13.49.1.144 CFE_TBL_ERR_INVALID_NAME

```
#define CFE_TBL_ERR_INVALID_NAME ((int32)0xcc000002)
```

The calling Application attempted to register a table whose name length exceeded the platform configuration value of CFE_MISSION_TBL_MAX_NAME_LENGTH or was zero characters long.

Definition at line 967 of file cfe_error.h.

13.49.1.145 CFE_TBL_ERR_INVALID_OPTIONS

```
#define CFE_TBL_ERR_INVALID_OPTIONS ((int32)0xcc000025)
```

The calling Application has used an illegal combination of table options. A summary of the illegal combinations are as follows:

#CFE_TBL_OPT_USR_DEF_ADDR cannot be combined with any of the following:

1. CFE_TBL_OPT_DBL_BUFFER
2. CFE_TBL_OPT_LOAD_DUMP
3. CFE_TBL_OPT_CRITICAL

#CFE_TBL_OPT_DBL_BUFFER cannot be combined with the following:

1. CFE_TBL_OPT_USR_DEF_ADDR
2. CFE_TBL_OPT_DUMP_ONLY

Definition at line 1239 of file cfe_error.h.

13.49.1.146 CFE_TBL_ERR_INVALID_SIZE

```
#define CFE_TBL_ERR_INVALID_SIZE ((int32)0xcc000003)
```

The calling Application attempted to register a table: a) that was a double buffered table with size greater than CFE_↩ PLATFORM_TBL_MAX_DBL_TABLE_SIZE b) that was a single buffered table with size greater than CFE_PLATFO↩ RM_TBL_MAX_SNGL_TABLE_SIZE c) that had a size of zero

Definition at line 976 of file cfe_error.h.

13.49.1.147 CFE_TBL_ERR_LOAD_IN_PROGRESS

```
#define CFE_TBL_ERR_LOAD_IN_PROGRESS ((int32)0xcc000012)
```

The calling Application called CFE_TBL_Load when another Application was trying to load the table.

Definition at line 1090 of file cfe_error.h.

13.49.1.148 CFE_TBL_ERR_LOAD_INCOMPLETE

```
#define CFE_TBL_ERR_LOAD_INCOMPLETE ((int32)0xcc000021)
```

The calling Application tried to load a table file whose header claimed the load was larger than what was actually read from the file.

Definition at line 1195 of file cfe_error.h.

13.49.1.149 CFE_TBL_ERR_NEVER_LOADED

```
#define CFE_TBL_ERR_NEVER_LOADED ((int32)0xcc000005)
```

This is an error indicating that the table has never been loaded from either a file or a copy from a block of memory so the contents that the returned pointer is pointing to are zeros. **NOTE: Unlike other most other errors, this error condition still returns a valid table pointer. This pointer must be released with the CFE_TBL_ReleaseAddress API before the table can be loaded with data.**

Definition at line 993 of file cfe_error.h.

13.49.1.150 CFE_TBL_ERR_NO_ACCESS

```
#define CFE_TBL_ERR_NO_ACCESS ((int32)0xcc000008)
```

The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.

Definition at line 1015 of file cfe_error.h.

13.49.1.151 CFE_TBL_ERR_NO_BUFFER_AVAIL

```
#define CFE_TBL_ERR_NO_BUFFER_AVAIL ((int32)0xcc00000F)
```

The calling Application has tried to allocate a working buffer but none were available.

Definition at line 1069 of file cfe_error.h.

13.49.1.152 CFE_TBL_ERR_NO_STD_HEADER

```
#define CFE_TBL_ERR_NO_STD_HEADER ((int32)0xcc00001D)
```

The calling Application tried to access a table file whose standard cFE File Header was the wrong size, etc.

Definition at line 1164 of file cfe_error.h.

13.49.1.153 CFE_TBL_ERR_NO_TBL_HEADER

```
#define CFE_TBL_ERR_NO_TBL_HEADER ((int32)0xcc00001E)
```

The calling Application tried to access a table file whose standard cFE Table File Header was the wrong size, etc.

Definition at line 1171 of file cfe_error.h.

13.49.1.154 CFE_TBL_ERR_PARTIAL_LOAD

```
#define CFE_TBL_ERR_PARTIAL_LOAD ((int32)0xcc000023)
```

The calling Application tried to load a table file whose header claimed the load did not start with the first byte and the table image had NEVER been loaded before. Partial loads are not allowed on uninitialized tables. It should be noted that CFE_TBL_WARN_SHORT_FILE also indicates a partial load.

Definition at line 1215 of file cfe_error.h.

13.49.1.155 CFE_TBL_ERR_REGISTRY_FULL

```
#define CFE_TBL_ERR_REGISTRY_FULL ((int32)0xcc000006)
```

An application attempted to create a table and the Table registry already contained CFE_PLATFORM_TBL_MAX_N↩ UM_TABLES in it.

Definition at line 1000 of file cfe_error.h.

13.49.1.156 CFE_TBL_ERR_UNREGISTERED

```
#define CFE_TBL_ERR_UNREGISTERED ((int32)0xcc000009)
```

The calling application is trying to access a table that has been unregistered.

Definition at line 1022 of file cfe_error.h.

13.49.1.157 CFE_TBL_INFO_DUMP_PENDING

```
#define CFE_TBL_INFO_DUMP_PENDING ((int32)0x4c000024)
```

The calling Application should call CFE_TBL_Manage for the specified table. The ground has requested a dump of the Dump-Only table and needs to synchronize with the owning application.

Definition at line 1224 of file cfe_error.h.

13.49.1.158 CFE_TBL_INFO_NO_UPDATE_PENDING

```
#define CFE_TBL_INFO_NO_UPDATE_PENDING ((int32)0x4c000017)
```

The calling Application has attempted to update a table without a pending load.

Definition at line 1126 of file cfe_error.h.

13.49.1.159 CFE_TBL_INFO_NO_VALIDATION_PENDING

```
#define CFE_TBL_INFO_NO_VALIDATION_PENDING ((int32)0x4c00001A)
```

The calling Application tried to validate a table that did not have a validation request pending.

Definition at line 1144 of file cfe_error.h.

13.49.1.160 CFE_TBL_INFO_RECOVERED_TBL

```
#define CFE_TBL_INFO_RECOVERED_TBL ((int32)0x4c000027)
```

The calling Application registered a critical table whose previous contents were discovered in the Critical Data Store. The discovered contents were copied back into the newly registered table as the table's initial contents.
NOTE: In this situation, the contents of the table are **NOT** validated using the table's validation function.

Definition at line 1261 of file cfe_error.h.

13.49.1.161 CFE_TBL_INFO_TABLE_LOCKED

```
#define CFE_TBL_INFO_TABLE_LOCKED ((int32)0x4c000018)
```

The calling Application tried to update a table that is locked by another user.

Definition at line 1132 of file cfe_error.h.

13.49.1.162 CFE_TBL_INFO_UPDATE_PENDING

```
#define CFE_TBL_INFO_UPDATE_PENDING ((int32)0x4c000004)
```

The calling Application has identified a table that has a load pending.

Definition at line 982 of file cfe_error.h.

13.49.1.163 CFE_TBL_INFO_UPDATED

```
#define CFE_TBL_INFO_UPDATED ((int32)0x4c00000E)
```

The calling Application has identified a table that has been updated.
NOTE: This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.

Definition at line 1062 of file cfe_error.h.

13.49.1.164 CFE_TBL_INFO_VALIDATION_PENDING

```
#define CFE_TBL_INFO_VALIDATION_PENDING ((int32)0x4c000019)
```

The calling Application should call CFE_TBL_Validate for the specified table.

Definition at line 1138 of file cfe_error.h.

13.49.1.165 CFE_TBL_MESSAGE_ERROR

```
#define CFE_TBL_MESSAGE_ERROR ((int32)0xcc00002a)
```

Error code indicating that the TBL command was not processed successfully and that the error counter should be incremented.

Definition at line 1287 of file cfe_error.h.

13.49.1.166 CFE_TBL_NOT_IMPLEMENTED

```
#define CFE_TBL_NOT_IMPLEMENTED ((int32)0xcc00ffff)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 1297 of file cfe_error.h.

13.49.1.167 CFE_TBL_WARN_DUPLICATE

```
#define CFE_TBL_WARN_DUPLICATE ((int32)0x4c000007)
```

This is an error that the registration is trying to replace an existing table with the same name. The previous table stays in place and the new table is rejected.

Definition at line 1008 of file cfe_error.h.

13.49.1.168 CFE_TBL_WARN_NOT_CRITICAL

```
#define CFE_TBL_WARN_NOT_CRITICAL ((int32)0x4c000026)
```

The calling Application attempted to register a table as "Critical". Table Services failed to create an appropriate Critical Data Store (See System Log for reason) to save the table contents. The table will be treated as a normal table from now on.

Definition at line 1249 of file cfe_error.h.

13.49.1.169 CFE_TBL_WARN_PARTIAL_LOAD

```
#define CFE_TBL_WARN_PARTIAL_LOAD ((int32)0x4c000022)
```

The calling Application tried to load a table file whose header claimed the load did not start with the first byteIt should be noted that CFE_TBL_WARN_SHORT_FILE also indicates a partial load.

Definition at line 1204 of file cfe_error.h.

13.49.1.170 CFE_TBL_WARN_SHORT_FILE

```
#define CFE_TBL_WARN_SHORT_FILE ((int32)0x4c000015)
```

The calling Application called CFE_TBL_Load with a filename that specified a file that started with the first byte of the table but contained less data than the size of the table. It should be noted that CFE_TBL_WARN_PARTIAL_LOAD also indicates a partial load (one that starts at a non-zero offset).

Definition at line 1113 of file cfe_error.h.

13.49.1.171 CFE_TIME_CALLBACK_NOT_REGISTERED

```
#define CFE_TIME_CALLBACK_NOT_REGISTERED ((int32)0xce000004)
```

An attempt to unregister a cFE Time Services Synchronization callback has failed because the specified callback function was not located in the Synchronization Callback Registry.

Definition at line 1351 of file cfe_error.h.

13.49.1.172 CFE_TIME_INTERNAL_ONLY

```
#define CFE_TIME_INTERNAL_ONLY ((int32)0xce000001)
```

One of the TIME Services API functions to set the time with data from an external time source has been called, but TIME Services has been commanded to not accept external time data. However, the command is still a signal for the Time Server to generate a "time at the tone" command packet using internal data.

Definition at line 1321 of file cfe_error.h.

13.49.1.173 CFE_TIME_NOT_IMPLEMENTED

```
#define CFE_TIME_NOT_IMPLEMENTED ((int32)0xce00ffff)
```

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 1311 of file cfe_error.h.

13.49.1.174 CFE_TIME_OUT_OF_RANGE

```
#define CFE_TIME_OUT_OF_RANGE ((int32)0xce000002)
```

One of the TIME Services API functions to set the time with data from an external time source has been called, but TIME Services has determined that the new time data is invalid. However, the command is still a signal for the Time Server to generate a "time at the tone" command packet using internal data.

Note that the test for invalid time update data only occurs if TIME Services has previously been commanded to set the clock state to "valid".

Definition at line 1334 of file cfe_error.h.

13.49.1.175 CFE_TIME_SERVICE

```
#define CFE_TIME_SERVICE ((int32)0x0e000000)
```

Definition at line 105 of file cfe_error.h.

13.49.1.176 CFE_TIME_TOO_MANY_SYNCH_CALLBACKS

```
#define CFE_TIME_TOO_MANY_SYNCH_CALLBACKS ((int32)0xce000003)
```

An attempt to register too many cFE Time Services Synchronization callbacks has been made. Only one callback function is allowed per application. It is expected that the application itself will distribute the single callback to child threads as needed.

Definition at line 1343 of file cfe_error.h.

13.50 cfe/fsw/cfe-core/src/inc/cfe_es.h File Reference

```
#include "cfe_es_extern_typedefs.h"
#include "cfe_mission_cfg.h"
#include "cfe_perfids.h"
```

Data Structures

- struct CFE_ES_AppInfo_t
- struct CFE_ES_TaskInfo_t
- struct CFE_ES_BlockStats_t
- struct CFE_ES_MemPoolStats_t
- struct CFE_ES_CDSRegDumpRec_t
- union CFE_ES_PoolAlign_t

Macros

- #define OS_PRINTF(m, n)
- #define CFE_ES_DBIT(x) (1L << (x)) /* Places a one at bit positions 0 thru 31 */
- #define CFE_ES_DTEST(i, x) (((i) & CFE_ES_DBIT(x)) != 0) /* true iff bit x of i is set */
- #define CFE_ES_TEST_LONG_MASK(m, s) (CFE_ES_DTEST(m[(s)/32],(s)%32)) /* Test a bit within an array of 32-bit integers. */
- #define CFE_ES_MAX_MEMPOOL_BLOCK_SIZES 17
- #define CFE_ES_NO_MUTEX 0

  Indicates that the memory pool selection will not use a semaphore.
- #define CFE_ES_USE_MUTEX 1

  Indicates that the memory pool selection will use a semaphore.
- #define CFE_ES_PROCESSOR_RESET CFE_PSP_RST_TYPE_PROCESSOR
- #define CFE_ES_POWERON_RESET CFE_PSP_RST_TYPE_POWERON
- #define CFE_ES_POWER_CYCLE CFE_PSP_RST_SUBTYPE_POWER_CYCLE
- #define CFE_ES_PUSH_BUTTON CFE_PSP_RST_SUBTYPE_PUSH_BUTTON
- #define CFE_ES_HW_SPECIAL_COMMAND CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND
- #define CFE_ES_HW_WATCHDOG CFE_PSP_RST_SUBTYPE_HW_WATCHDOG
- #define CFE_ES_RESET_COMMAND CFE_PSP_RST_SUBTYPE_RESET_COMMAND
- #define CFE_ES_EXCEPTION CFE_PSP_RST_SUBTYPE_EXCEPTION
- #define CFE_ES_UNDEFINED_RESET CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET
- #define CFE_ES_HWDEBUG_RESET CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET
- #define CFE_ES_BANKSWITCH_RESET CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET
- #define CFE_ES_SYSTEM_STATE_UNDEFINED CFE_ES_SystemState_UNDEFINED
- #define CFE_ES_SYSTEM_STATE_EARLY_INIT CFE_ES_SystemState_EARLY_INIT
- #define CFE_ES_SYSTEM_STATE_CORE_STARTUP CFE_ES_SystemState_CORE_STARTUP
- #define CFE_ES_SYSTEM_STATE_CORE_READY CFE_ES_SystemState_CORE_READY
- #define CFE_ES_SYSTEM_STATE_APPS_INIT CFE_ES_SystemState_APPS_INIT
- #define CFE_ES_SYSTEM_STATE_OPERATIONAL CFE_ES_SystemState_OPERATIONAL
- #define CFE_ES_SYSTEM_STATE_SHUTDOWN CFE_ES_SystemState_SHUTDOWN
- #define CFE_ES_APP_RUN CFE_ES_RunStatus_APP_RUN
- #define CFE_ES_APP_EXIT CFE_ES_RunStatus_APP_EXIT
- #define CFE_ES_APP_ERROR CFE_ES_RunStatus_APP_ERROR

- #define CFE_ES_SYS_EXCEPTION CFE_ES_RunStatus_SYS_EXCEPTION
- #define CFE_ES_SYS_RESTART CFE_ES_RunStatus_SYS_RESTART
- #define CFE_ES_SYS_RELOAD CFE_ES_RunStatus_SYS_RELOAD
- #define CFE_ES_SYS_DELETE CFE_ES_RunStatus_SYS_DELETE
- #define CFE_ES_CORE_APP_INIT_ERROR CFE_ES_RunStatus_CORE_APP_INIT_ERROR
- #define CFE_ES_CORE_APP_RUNTIME_ERROR CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR
- #define CFE_ES_APP_STATE_UNDEFINED CFE_ES_AppState_UNDEFINED
- #define CFE_ES_APP_STATE_EARLY_INIT CFE_ES_AppState_EARLY_INIT
- #define CFE_ES_APP_STATE_LATE_INIT CFE_ES_AppState_LATE_INIT
- #define CFE_ES_APP_STATE_RUNNING CFE_ES_AppState_RUNNING
- #define CFE_ES_APP_STATE_WAITING CFE_ES_AppState_WAITING
- #define CFE_ES_APP_STATE_STOPPED CFE_ES_AppState_STOPPED
- #define CFE_ES_APP_TYPE_CORE CFE_ES_AppType_CORE
- #define CFE_ES_APP_TYPE_EXTERNAL CFE_ES_AppType_EXTERNAL
- #define CFE_ES_LOG_DISCARD CFE_ES_LogMode_DISCARD
- #define CFE_ES_LOG_OVERWRITE CFE_ES_LogMode_OVERWRITE
- #define CFE_ES_APP_EXCEPTION_RESTART_APP CFE_ES_ExceptionAction_RESTART_APP
- #define CFE_ES_APP_EXCEPTION_PROC_RESTART CFE_ES_ExceptionAction_PROC_RESTART
- #define CFE_ES_CORE_LOG_ENTRY CFE_ES_LogEntryType_CORE
- #define CFE_ES_APPLICATION_LOG_ENTRY CFE_ES_LogEntryType_APPLICATION
- #define CFE_ES_STATIC_POOL_TYPE(size) union { CFE_ES_PoolAlign_t Align; uint8 Data[size]; }
- #define CFE_ES_PerfLogEntry(id) (CFE_ES_PerfLogAdd(id, 0))

  Entry marker for use with Software Performance Analysis Tool.
- #define CFE_ES_PerfLogExit(id) (CFE_ES_PerfLogAdd(id, 1))

  Exit marker for use with Software Performance Analysis Tool.

Reset Type extensions

- #define CFE_ES_APP_RESTART CFE_PSP_RST_TYPE_MAX

Critical Data Store Macros

- #define CFE_ES_CDS_MAX_FULL_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + OS↩ _MAX_API_NAME + 2)
- #define CFE_ES_CDS_BAD_HANDLE (CFE_ES_CDSHandle_t) 0xFFFF

Typedefs

- typedef cpuaddr CFE_ES_MemHandle_t

  Data type used to hold Handles of Memory Pools created via CFE_ES_PoolCreate and CFE_ES_PoolCreateNoSem.
- typedef cpuaddr CFE_ES_CDSHandle_t

  Data type used to hold Handles of Critical Data Stores. See CFE_ES_RegisterCDS.
- typedef void(∗ CFE_ES_ChildTaskMainFuncPtr_t) (void)

  Required Prototype of Child Task Main Functions.
- typedef int32(∗ CFE_ES_LibraryEntryFuncPtr_t) (uint32 LibId)

  Required Prototype of Library Initialization Functions.

Functions

- void CFE_ES_Main (uint32 StartType, uint32 StartSubtype, uint32 ModeId, const char *StartFilePath)

  cFE Main Entry Point used by Board Support Package to start cFE
- int32 CFE_ES_GetResetType (uint32 *ResetSubtypePtr)

  Return the most recent Reset Type.
- int32 CFE_ES_ResetCFE (uint32 ResetType)

  Reset the cFE Core and all cFE Applications.
- int32 CFE_ES_RestartApp (uint32 AppID)

  Restart a single cFE Application.
- int32 CFE_ES_ReloadApp (uint32 AppID, const char *AppFileName)

  Reload a single cFE Application.
- int32 CFE_ES_DeleteApp (uint32 AppID)

  Delete a cFE Application.
- void CFE_ES_ExitApp (uint32 ExitStatus)

  Exit a cFE Application.
- bool CFE_ES_RunLoop (uint32 *ExitStatus)

  Check for Exit, Restart, or Reload commands.
- int32 CFE_ES_WaitForSystemState (uint32 MinSystemState, uint32 TimeOutMilliseconds)

  Allow an Application to Wait for a minimum global system state.
- void CFE_ES_WaitForStartupSync (uint32 TimeOutMilliseconds)

  Allow an Application to Wait for the "OPERATIONAL" global system state.
- int32 CFE_ES_RegisterApp (void)

  Registers a cFE Application with the Executive Services.
- int32 CFE_ES_GetAppID (uint32 *AppIdPtr)

  Get an Application ID for the calling Application.
- int32 CFE_ES_GetAppIDByName (uint32 *AppIdPtr, const char *AppName)

  Get an Application ID associated with a specified Application name.
- int32 CFE_ES_GetAppName (char *AppName, uint32 AppId, uint32 BufferLength)

  Get an Application name for a specified Application ID.
- int32 CFE_ES_GetAppInfo (CFE_ES_AppInfo_t *AppInfo, uint32 AppId)

  Get Application Information given a specified App ID.
- int32 CFE_ES_GetTaskInfo (CFE_ES_TaskInfo_t *TaskInfo, uint32 TaskId)

  Get Task Information given a specified Task ID.
- int32 CFE_ES_RegisterChildTask (void)

  Registers a cFE Child task associated with a cFE Application.
- int32 CFE_ES_CreateChildTask (uint32 *TaskIdPtr, const char *TaskName, CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr, uint32 *StackPtr, uint32 StackSize, uint32 Priority, uint32 Flags)

  Creates a new task under an existing Application.
- int32 CFE_ES_DeleteChildTask (uint32 TaskId)

  Deletes a task under an existing Application.
- void CFE_ES_ExitChildTask (void)

  Exits a child task.
- void CFE_ES_IncrementTaskCounter (void)

  Increments the execution counter for the calling task.
- int32 CFE_ES_WriteToSysLog (const char *SpecStringPtr,...) OS_PRINTF(1

  Write a string to the cFE System Log.

- int32 uint32 CFE_ES_CalculateCRC (const void *DataPtr, uint32 DataLength, uint32 InputCRC, uint32 TypeCRC)

  Calculate a CRC on a block of memory.
- int32 CFE_ES_RegisterCDS (CFE_ES_CDSHandle_t *HandlePtr, int32 BlockSize, const char *Name)

  Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
- int32 CFE_ES_CopyToCDS (CFE_ES_CDSHandle_t Handle, void *DataToCopy)

  Save a block of data in the Critical Data Store (CDS)
- int32 CFE_ES_RestoreFromCDS (void *RestoreToMemory, CFE_ES_CDSHandle_t Handle)

  Recover a block of data from the Critical Data Store (CDS)
- int32 CFE_ES_PoolCreateNoSem (CFE_ES_MemHandle_t *HandlePtr, uint8 *MemPtr, uint32 Size)

  Initializes a memory pool created by an application without using a semaphore during processing.
- int32 CFE_ES_PoolCreate (CFE_ES_MemHandle_t *HandlePtr, uint8 *MemPtr, uint32 Size)

  Initializes a memory pool created by an application while using a semaphore during processing.
- int32 CFE_ES_PoolCreateEx (CFE_ES_MemHandle_t *HandlePtr, uint8 *MemPtr, uint32 Size, uint32 Num↩ BlockSizes, uint32 *BlockSizes, uint16 UseMutex)

  Initializes a memory pool created by an application with application specified block sizes.
- int32 CFE_ES_GetPoolBuf (uint32 **BufPtr, CFE_ES_MemHandle_t HandlePtr, uint32 Size)

  Gets a buffer from the memory pool created by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
- int32 CFE_ES_GetPoolBufInfo (CFE_ES_MemHandle_t HandlePtr, uint32 *BufPtr)

  Gets info on a buffer previously allocated via CFE_ES_GetPoolBuf.
- int32 CFE_ES_PutPoolBuf (CFE_ES_MemHandle_t HandlePtr, uint32 *BufPtr)

  Releases a buffer from the memory pool that was previously allocated via CFE_ES_GetPoolBuf.
- int32 CFE_ES_GetMemPoolStats (CFE_ES_MemPoolStats_t *BufPtr, CFE_ES_MemHandle_t Handle)

  Extracts the statistics maintained by the memory pool software.
- void CFE_ES_PerfLogAdd (uint32 Marker, uint32 EntryExit)

  Function called by CFE_ES_PerfLogEntry and CFE_ES_PerfLogExit macros.
- int32 CFE_ES_RegisterGenCounter (uint32 *CounterIdPtr, const char *CounterName)

  Register a generic counter.
- int32 CFE_ES_DeleteGenCounter (uint32 CounterId)

  Delete a generic counter.
- int32 CFE_ES_IncrementGenCounter (uint32 CounterId)

  Increments the specified generic counter.
- int32 CFE_ES_SetGenCount (uint32 CounterId, uint32 Count)

  Set the specified generic counter.
- int32 CFE_ES_GetGenCount (uint32 CounterId, uint32 *Count)

  Get the specified generic counter count.
- int32 CFE_ES_GetGenCounterIDByName (uint32 *CounterIdPtr, const char *CounterName)

  Get the Id associated with a generic counter name.
- void CFE_ES_ProcessCoreException (uint32 HostTaskId, const char *ReasonString, const uint32 *Context↩ Pointer, uint32 ContextSize)

  Process an exception detected by the underlying OS/PSP.

13.50.1 Macro Definition Documentation

13.50.1.1 CFE_ES_APP_ERROR

```
#define CFE_ES_APP_ERROR CFE_ES_RunStatus_APP_ERROR
```

Definition at line 143 of file cfe_es.h.

13.50.1.2 CFE_ES_APP_EXCEPTION_PROC_RESTART

```
#define CFE_ES_APP_EXCEPTION_PROC_RESTART CFE_ES_ExceptionAction_PROC_RESTART
```

Definition at line 177 of file cfe_es.h.

13.50.1.3 CFE_ES_APP_EXCEPTION_RESTART_APP

```
#define CFE_ES_APP_EXCEPTION_RESTART_APP CFE_ES_ExceptionAction_RESTART_APP
```

Definition at line 176 of file cfe_es.h.

13.50.1.4 CFE_ES_APP_EXIT

```
#define CFE_ES_APP_EXIT CFE_ES_RunStatus_APP_EXIT
```

Definition at line 142 of file cfe_es.h.

13.50.1.5 CFE_ES_APP_RESTART

```
#define CFE_ES_APP_RESTART CFE_PSP_RST_TYPE_MAX
```

Application only was reset (extend the PSP enumeration here)

Definition at line 80 of file cfe_es.h.

Referenced by CFE_ES_ProcessCoreException().

13.50.1.6 CFE_ES_APP_RUN

```
#define CFE_ES_APP_RUN CFE_ES_RunStatus_APP_RUN
```

Definition at line 141 of file cfe_es.h.

13.50.1.7 CFE_ES_APP_STATE_EARLY_INIT

```
#define CFE_ES_APP_STATE_EARLY_INIT CFE_ES_AppState_EARLY_INIT
```

Definition at line 155 of file cfe_es.h.

13.50.1.8 CFE_ES_APP_STATE_LATE_INIT

```
#define CFE_ES_APP_STATE_LATE_INIT CFE_ES_AppState_LATE_INIT
```

Definition at line 156 of file cfe_es.h.

13.50.1.9 CFE_ES_APP_STATE_RUNNING

```
#define CFE_ES_APP_STATE_RUNNING CFE_ES_AppState_RUNNING
```

Definition at line 157 of file cfe_es.h.

13.50.1.10 CFE_ES_APP_STATE_STOPPED

```
#define CFE_ES_APP_STATE_STOPPED CFE_ES_AppState_STOPPED
```

Definition at line 159 of file cfe_es.h.

13.50.1.11 CFE_ES_APP_STATE_UNDEFINED

```
#define CFE_ES_APP_STATE_UNDEFINED CFE_ES_AppState_UNDEFINED
```

Definition at line 154 of file cfe_es.h.

13.50.1.12 CFE_ES_APP_STATE_WAITING

```
#define CFE_ES_APP_STATE_WAITING CFE_ES_AppState_WAITING
```

Definition at line 158 of file cfe_es.h.

13.50.1.13 CFE_ES_APP_TYPE_CORE

```
#define CFE_ES_APP_TYPE_CORE CFE_ES_AppType_CORE
```

Definition at line 164 of file cfe_es.h.

13.50.1.14 CFE_ES_APP_TYPE_EXTERNAL

```
#define CFE_ES_APP_TYPE_EXTERNAL CFE_ES_AppType_EXTERNAL
```

Definition at line 165 of file cfe_es.h.

13.50.1.15 CFE_ES_APPLICATION_LOG_ENTRY

```
#define CFE_ES_APPLICATION_LOG_ENTRY CFE_ES_LogEntryType_APPLICATION
```

Definition at line 183 of file cfe_es.h.

13.50.1.16 CFE_ES_BANKSWITCH_RESET

```
#define CFE_ES_BANKSWITCH_RESET CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET
```

Definition at line 125 of file cfe_es.h.

13.50.1.17 CFE_ES_CDS_BAD_HANDLE

```
#define CFE_ES_CDS_BAD_HANDLE (CFE_ES_CDSHandle_t) 0xFFFF
```

Definition at line 90 of file cfe_es.h.

Referenced by CFE_ES_RegisterCDS().

13.50.1.18 CFE_ES_CDS_MAX_FULL_NAME_LEN

```
#define CFE_ES_CDS_MAX_FULL_NAME_LEN (CFE_MISSION_ES_CDS_MAX_NAME_LENGTH + OS_MAX_API_NAME + 2)
```

Maximum length allowed for CDS name.
NOTE: "+2" is for NULL Character and "." (i.e. - "AppName.CDSName")

Definition at line 88 of file cfe_es.h.

Referenced by CFE_ES_DeleteCDSCmd(), CFE_ES_RegisterCDS(), and CFE_ES_RegisterCDSEx().

13.50.1.19 CFE_ES_CORE_APP_INIT_ERROR

```
#define CFE_ES_CORE_APP_INIT_ERROR CFE_ES_RunStatus_CORE_APP_INIT_ERROR
```

Definition at line 148 of file cfe_es.h.

13.50.1.20 CFE_ES_CORE_APP_RUNTIME_ERROR

```
#define CFE_ES_CORE_APP_RUNTIME_ERROR CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR
```

Definition at line 149 of file cfe_es.h.

13.50.1.21 CFE_ES_CORE_LOG_ENTRY

```
#define CFE_ES_CORE_LOG_ENTRY CFE_ES_LogEntryType_CORE
```

Definition at line 182 of file cfe_es.h.

13.50.1.22 CFE_ES_DBIT

```
#define CFE_ES_DBIT(
              x ) (1L << (x)) /* Places a one at bit positions 0 thru 31 */
```

Definition at line 60 of file cfe_es.h.

13.50.1.23 CFE_ES_DTEST

```
#define CFE_ES_DTEST(
              i,
              x ) (((i) & CFE_ES_DBIT(x)) != 0) /* true iff bit x of i is set */
```

Definition at line 61 of file cfe_es.h.

13.50.1.24 CFE_ES_EXCEPTION

```
#define CFE_ES_EXCEPTION CFE_PSP_RST_SUBTYPE_EXCEPTION
```

Definition at line 122 of file cfe_es.h.

13.50.1.25 CFE_ES_HW_SPECIAL_COMMAND

```
#define CFE_ES_HW_SPECIAL_COMMAND CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND
```

Definition at line 119 of file cfe_es.h.

13.50.1.26 CFE_ES_HW_WATCHDOG

```
#define CFE_ES_HW_WATCHDOG CFE_PSP_RST_SUBTYPE_HW_WATCHDOG
```

Definition at line 120 of file cfe_es.h.

13.50.1.27 CFE_ES_HWDEBUG_RESET

```
#define CFE_ES_HWDEBUG_RESET CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET
```

Definition at line 124 of file cfe_es.h.

13.50.1.28 CFE_ES_LOG_DISCARD

```
#define CFE_ES_LOG_DISCARD CFE_ES_LogMode_DISCARD
```

Definition at line 170 of file cfe_es.h.

13.50.1.29 CFE_ES_LOG_OVERWRITE

```
#define CFE_ES_LOG_OVERWRITE CFE_ES_LogMode_OVERWRITE
```

Definition at line 171 of file cfe_es.h.

13.50.1.30 CFE_ES_MAX_MEMPOOL_BLOCK_SIZES

```
#define CFE_ES_MAX_MEMPOOL_BLOCK_SIZES 17
```

Max number of size divisions allowed in a memory pool

Definition at line 63 of file cfe_es.h.

Referenced by CFE_ES_GetBlockSize(), CFE_ES_GetMemPoolStats(), CFE_ES_PoolCreate(), CFE_ES_Pool↩ CreateEx(), CFE_ES_PoolCreateNoSem(), and CFE_SB_InitBuffers().

13.50.1.31 CFE_ES_NO_MUTEX

```
#define CFE_ES_NO_MUTEX 0
```

Definition at line 93 of file cfe_es.h.

Referenced by CFE_ES_PoolCreateEx(), CFE_ES_PoolCreateNoSem(), and CFE_SB_InitBuffers().

13.50.1.32 CFE_ES_PerfLogEntry

```
#define CFE_ES_PerfLogEntry(
              id ) (CFE_ES_PerfLogAdd(id, 0))
```

Description

This macro logs the entry or start event/marker for the specified entry `id`. This macro, in conjunction with the CFE_ES_PerfLogExit, is used by the Software Performance Analysis tool (see section 5.15).

Assumptions, External Events, and Notes:

None

Parameters

in	*id*	Identifier of the specific event or marker.

See also

CFE_ES_PerfLogExit, CFE_ES_PerfLogAdd

Definition at line 1319 of file cfe_es.h.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), CFE_SB_SendMsgFull(), and CFE_SB_TaskMain().

13.50.1.33 CFE_ES_PerfLogExit

```
#define CFE_ES_PerfLogExit(
              id ) (CFE_ES_PerfLogAdd(id, 1))
```

Description

This macro logs the exit or end event/marker for the specified entry `id`. This macro, in conjunction with the CFE_ES_PerfLogEntry, is used by the Software Performance Analysis tool (see section 5.15).

Assumptions, External Events, and Notes:

None

Parameters

in	*id*	Identifier of the specific event or marker.

See also

CFE_ES_PerfLogEntry, CFE_ES_PerfLogAdd

Definition at line 1338 of file cfe_es.h.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), CFE_SB_SendMsgFull(), and CFE_SB_TaskMain().

13.50.1.34 CFE_ES_POWER_CYCLE

```
#define CFE_ES_POWER_CYCLE CFE_PSP_RST_SUBTYPE_POWER_CYCLE
```

Definition at line 117 of file cfe_es.h.

13.50.1.35 CFE_ES_POWERON_RESET

```
#define CFE_ES_POWERON_RESET CFE_PSP_RST_TYPE_POWERON
```

Definition at line 115 of file cfe_es.h.

13.50.1.36 CFE_ES_PROCESSOR_RESET

```
#define CFE_ES_PROCESSOR_RESET CFE_PSP_RST_TYPE_PROCESSOR
```

Definition at line 114 of file cfe_es.h.

13.50.1.37 CFE_ES_PUSH_BUTTON

```
#define CFE_ES_PUSH_BUTTON CFE_PSP_RST_SUBTYPE_PUSH_BUTTON
```

Definition at line 118 of file cfe_es.h.

13.50.1.38 CFE_ES_RESET_COMMAND

```
#define CFE_ES_RESET_COMMAND CFE_PSP_RST_SUBTYPE_RESET_COMMAND
```

Definition at line 121 of file cfe_es.h.

13.50.1.39 CFE_ES_STATIC_POOL_TYPE

```
#define CFE_ES_STATIC_POOL_TYPE(
              size ) union { CFE_ES_PoolAlign_t Align; uint8 Data[size]; }
```

A macro to help instantiate static memory pools that are correctly aligned. This resolves to a union type that contains a member called "Data" that will be correctly aligned to be a memory pool and sized according to the argument.

Definition at line 330 of file cfe_es.h.

Referenced by CFE_ES_PoolCreateEx().

13.50.1.40 CFE_ES_SYS_DELETE

```
#define CFE_ES_SYS_DELETE CFE_ES_RunStatus_SYS_DELETE
```

Definition at line 147 of file cfe_es.h.

13.50.1.41 CFE_ES_SYS_EXCEPTION

```
#define CFE_ES_SYS_EXCEPTION CFE_ES_RunStatus_SYS_EXCEPTION
```

Definition at line 144 of file cfe_es.h.

13.50.1.42 CFE_ES_SYS_RELOAD

```
#define CFE_ES_SYS_RELOAD CFE_ES_RunStatus_SYS_RELOAD
```

Definition at line 146 of file cfe_es.h.

13.50.1.43 CFE_ES_SYS_RESTART

```
#define CFE_ES_SYS_RESTART CFE_ES_RunStatus_SYS_RESTART
```

Definition at line 145 of file cfe_es.h.

13.50.1.44 CFE_ES_SYSTEM_STATE_APPS_INIT

```
#define CFE_ES_SYSTEM_STATE_APPS_INIT CFE_ES_SystemState_APPS_INIT
```

Definition at line 134 of file cfe_es.h.

13.50.1.45 CFE_ES_SYSTEM_STATE_CORE_READY

```
#define CFE_ES_SYSTEM_STATE_CORE_READY CFE_ES_SystemState_CORE_READY
```

Definition at line 133 of file cfe_es.h.

13.50.1.46 CFE_ES_SYSTEM_STATE_CORE_STARTUP

```
#define CFE_ES_SYSTEM_STATE_CORE_STARTUP CFE_ES_SystemState_CORE_STARTUP
```

Definition at line 132 of file cfe_es.h.

13.50.1.47 CFE_ES_SYSTEM_STATE_EARLY_INIT

```
#define CFE_ES_SYSTEM_STATE_EARLY_INIT CFE_ES_SystemState_EARLY_INIT
```

Definition at line 131 of file cfe_es.h.

13.50.1.48 CFE_ES_SYSTEM_STATE_OPERATIONAL

```
#define CFE_ES_SYSTEM_STATE_OPERATIONAL CFE_ES_SystemState_OPERATIONAL
```

Definition at line 135 of file cfe_es.h.

13.50.1.49 CFE_ES_SYSTEM_STATE_SHUTDOWN

```
#define CFE_ES_SYSTEM_STATE_SHUTDOWN CFE_ES_SystemState_SHUTDOWN
```

Definition at line 136 of file cfe_es.h.

13.50.1.50 CFE_ES_SYSTEM_STATE_UNDEFINED

```
#define CFE_ES_SYSTEM_STATE_UNDEFINED CFE_ES_SystemState_UNDEFINED
```

Definition at line 130 of file cfe_es.h.

13.50.1.51 CFE_ES_TEST_LONG_MASK

```
#define CFE_ES_TEST_LONG_MASK(
              m,
              s ) (CFE_ES_DTEST(m[(s)/32],(s)%32)) /* Test a bit within an array of 32-bit integers.
*/
```

Definition at line 62 of file cfe_es.h.

Referenced by CFE_ES_PerfLogAdd().

13.50.1.52 CFE_ES_UNDEFINED_RESET

```
#define CFE_ES_UNDEFINED_RESET CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET
```

Definition at line 123 of file cfe_es.h.

13.50.1.53 CFE_ES_USE_MUTEX

```
#define CFE_ES_USE_MUTEX 1
```

Definition at line 94 of file cfe_es.h.

Referenced by CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_PoolCreate(), CFE_ES_PoolCreateEx(), and CFE_ES_PutPoolBuf().

13.50.1.54 OS_PRINTF

```
#define OS_PRINTF(
              m,
              n )
```

Definition at line 57 of file cfe_es.h.

13.50.2 Typedef Documentation

13.50.2.1 CFE_ES_CDSHandle_t

```
typedef cpuaddr CFE_ES_CDSHandle_t
```

Definition at line 295 of file cfe_es.h.

13.50.2.2 CFE_ES_ChildTaskMainFuncPtr_t

```
typedef void(* CFE_ES_ChildTaskMainFuncPtr_t) (void)
```

Definition at line 309 of file cfe_es.h.

13.50.2.3 CFE_ES_LibraryEntryFuncPtr_t

```
typedef int32(* CFE_ES_LibraryEntryFuncPtr_t) (uint32 LibId)
```

Definition at line 310 of file cfe_es.h.

13.50.2.4 CFE_ES_MemHandle_t

```
typedef cpuaddr CFE_ES_MemHandle_t
```

Definition at line 196 of file cfe_es.h.

13.50.3 Function Documentation

13.50.3.1 CFE_ES_CalculateCRC()

```
int32 uint32 CFE_ES_CalculateCRC (
            const void * DataPtr,
            uint32 DataLength,
            uint32 InputCRC,
            uint32 TypeCRC )
```

Description

This routine calculates a cyclic redundancy check (CRC) on a block of memory. The CRC algorithm used is determined by the last parameter.

Assumptions, External Events, and Notes:

None

Parameters

in	*DataPtr*	Pointer to the base of the memory block.
in	*DataLength*	The number of bytes in the memory block.
in	*InputCRC*	A starting value for use in the CRC calculation. This parameter allows the user to calculate the CRC of non-contiguous blocks as a single value. Nominally, the user should set this value to zero.
in	*TypeCRC*	One of the following CRC algorithm selections: • `CFE_MISSION_ES_CRC_8` - (Not currently implemented) • `CFE_MISSION_ES_CRC_16` - a CRC-16 algorithm • `CFE_MISSION_ES_CRC_32` - (not currently implemented)

Returns

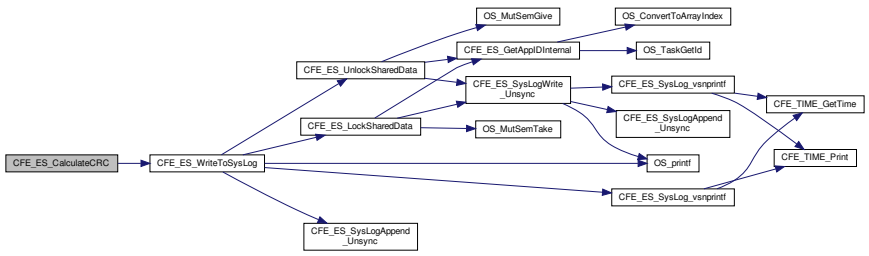
The result of the CRC calculation on the specified memory block.

Definition at line 1379 of file cfe_es_api.c.

References CFE_ES_WriteToSysLog(), CFE_MISSION_ES_CRC_16, CFE_MISSION_ES_CRC_32, and CFE_MIS↩SION_ES_CRC_8.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), and CFE_ES_TaskInit().

Here is the call graph for this function:



13.50.3.2 CFE_ES_CopyToCDS()

```
int32 CFE_ES_CopyToCDS (
            CFE_ES_CDSHandle_t Handle,
            void * DataToCopy )
```

Description

This routine copies a specified block of memory into the Critical Data Store that had been previously registered via CFE_ES_RegisterCDS. The block of memory to be copied must be at least as big as the size specified when registering the CDS.

Assumptions, External Events, and Notes:

None

Parameters

in	*Handle*	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS.
in	*DataToCopy*	A Pointer to the block of memory to be copied into the CDS.

OS_SUCCESS	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.
OS_ERROR	Problem with handle or a size mismatch

Returns

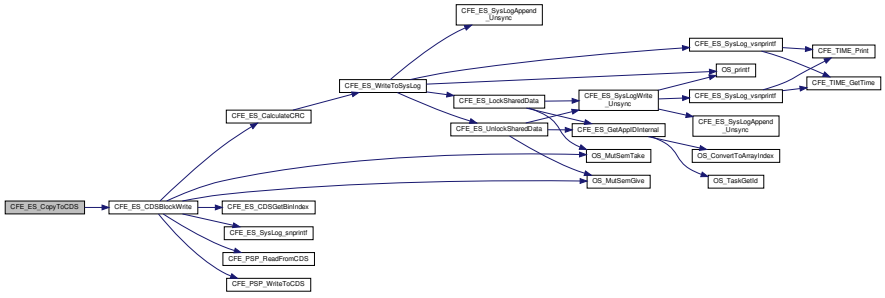
See also

CFE_ES_RegisterCDS, CFE_ES_RestoreFromCDS

Definition at line 1546 of file cfe_es_api.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDSBlockWrite(), CFE_ES_Global, CFE_ES_CDS_RegRec_t::↩ MemHandle, and CFE_ES_CDSVariables_t::Registry.

Here is the call graph for this function:



13.50.3.3 CFE_ES_CreateChildTask()

```
int32 CFE_ES_CreateChildTask (
            uint32 * TaskIdPtr,
            const char * TaskName,
            CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr,
            uint32 * StackPtr,
            uint32 StackSize,
            uint32 Priority,
            uint32 Flags )
```

Description

This routine creates a new task (a separate execution thread) owned by the calling Application.

Assumptions, External Events, and Notes:

None

Parameters

in	*TaskIdPtr*	A pointer to a variable that will be filled in with the new task's ID.
in	*TaskName*	A pointer to a string containing the desired name of the new task. This can be up to OS_MAX_API_NAME characters, including the trailing null.
in	*FunctionPtr*	A pointer to the function that will be spawned as a new task. This function must have the following signature: uint32 function(void). Input parameters for the new task are not supported.
in	*StackPtr*	A pointer to the location where the child task's stack pointer should start. NOTE: Not all underlying operating systems support this parameter.
in	*StackSize*	The number of bytes to allocate for the new task's stack.
in	*Priority*	The priority for the new task. Lower numbers are higher priority, with 0 being the highest priority. Applications cannot create tasks with a higher priority (lower number) than their own priority.
in	*Flags*	Reserved for future expansion.
out	*∗TaskIdPtr*	The Task ID of the newly created child task.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_CHILD_TASK_CREATE	There was an error creating a child task.

Returns

See also

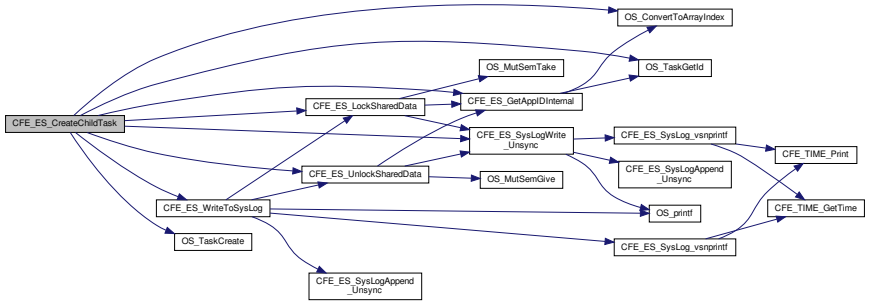
CFE_ES_RegisterChildTask, CFE_ES_DeleteChildTask, CFE_ES_ExitChildTask

Definition at line 985 of file cfe_es_api.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_Global_t::AppTable, CFE_ES_BAD_ARGUMENT, CFE_ES_ERR_CHILD_TASK_CREATE, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_SUCCESS, CFE_ES_MainTaskInfo_t::MainTaskId, NULL, OS_ConvertToArrayIndex(), OS_FP_ENABLED, OS_MAX_API_NAME, OS_SUCCESS, OS_TaskCreate(), OS_TaskGetId(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, CFE_ES_TaskRecord_t::TaskId, CFE_ES_AppRecord_t::TaskInfo, CFE_ES_TaskRecord_t::TaskName, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_StopPerfDataCmd().

Here is the call graph for this function:



13.50.3.4 CFE_ES_DeleteApp()

```
int32 CFE_ES_DeleteApp (
            uint32 AppID )
```

Description

This API causes a cFE Application to be stopped deleted.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppID*	Identifies the application to be reset.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

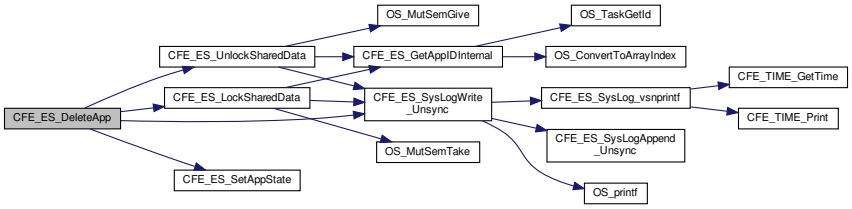
CFE_ES_RestartApp, CFE_ES_ReloadApp

Definition at line 333 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_RUNNING, CFE_ES_AppState_WAITING, CFE_ES_AppType_CORE, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_SYS_DELETE, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_APP_KILL_TIMEOUT, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::Name, CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_StopAppCmd().

Here is the call graph for this function:



13.50.3.5 CFE_ES_DeleteChildTask()

```
int32 CFE_ES_DeleteChildTask (
            uint32 TaskId )
```

Description

This routine deletes a task under an Application specified by the `TaskId` obtained when the child task was created using the CFE_ES_CreateChildTask API.

Assumptions, External Events, and Notes:

None

Parameters

in	*TaskId*	The task ID previously obtained when the Child Task was created with the CFE_ES_CreateChildTask API.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

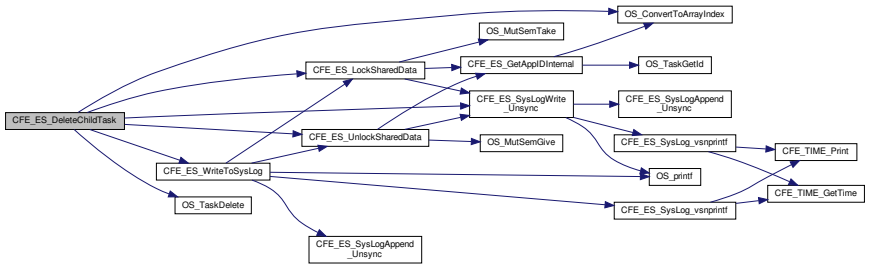
See also

CFE_ES_RegisterChildTask, CFE_ES_CreateChildTask, CFE_ES_ExitChildTask

Definition at line 1172 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_CHILD_TASK_DELETE, CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK, CFE_ES_ERR_TASKID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_MainTaskInfo_t::MainTaskId, OS_ConvertToArrayIndex(), OS_MAX_TASKS, OS_SUCCESS, OS_TaskDelete(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppRecord_t::TaskInfo, and CFE_ES_Global_t::TaskTable.

Here is the call graph for this function:



13.50.3.6 CFE_ES_DeleteGenCounter()

```
int32 CFE_ES_DeleteGenCounter (
            uint32 CounterId )
```

Description

This routine deletes a previously registered generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter Id of the newly created counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_IncrementGenCounter, CFE_ES_RegisterGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

Definition at line 1612 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.50.3.7 CFE_ES_ExitApp()

```
void CFE_ES_ExitApp (
            uint32 ExitStatus )
```

Description

This API is the "Exit Point" for the cFE application

Assumptions, External Events, and Notes:

None

Parameters

in	*ExitStatus*	.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

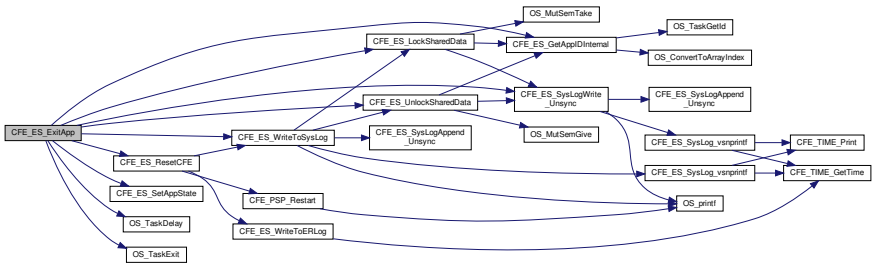
CFE_ES_RunLoop, CFE_ES_RegisterApp

Definition at line 375 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_Global_t::AppTable, CFE_ES_AppState_STOPPED, CFE_ES_AppType_CORE, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_ResetCFE(), CFE_ES_RunStatus_APP_ERROR, CFE_ES_RunStatus_APP_EXIT, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PSP_RST_TYPE_PROCESSOR, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::Name, OS_TaskDelay(), OS_TaskExit(), CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.50.3.8 CFE_ES_ExitChildTask()

```
void CFE_ES_ExitChildTask (
             void  )
```

Description

This routine allows the current executing child task to exit and be deleted by ES.

Assumptions, External Events, and Notes:

This function cannot be called from an Application's Main Task.

This function does not return a value, but if it does return at all, it is assumed that the Task was either unregistered or this function was called from a cFE Application's main task.

Returns

See also

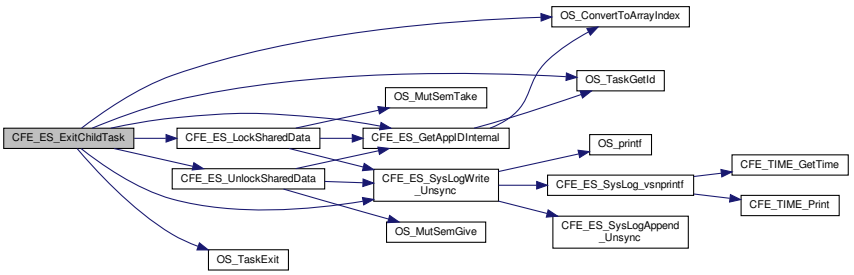
CFE_ES_RegisterChildTask, CFE_ES_CreateChildTask, CFE_ES_DeleteChildTask

Definition at line 1281 of file cfe_es_api.c.

References CFE_ES_Global_t::AppTable, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_SUCCESS, CFE_ES_MainTaskInfo_t::MainTaskId, OS_ConvertToArrayIndex(), OS_SUCCESS, OS_TaskExit(), OS_TaskGetId(), CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::RegisteredTasks, CFE_ES_AppRecord_t::TaskInfo, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_PerfLogDump().

Here is the call graph for this function:



13.50.3.9 CFE_ES_GetAppID()

```
int32 CFE_ES_GetAppID (
            uint32 * AppIdPtr )
```

Description

This routine retrieves the cFE Application ID for the calling Application.

Assumptions, External Events, and Notes:

NOTE: **All** tasks associated with the Application would return the same Application ID.

Parameters

in	*AppIdPtr*	Pointer to variable that is to receive the Application's ID.
out	∗*AppIdPtr*	Application ID of the calling Application.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

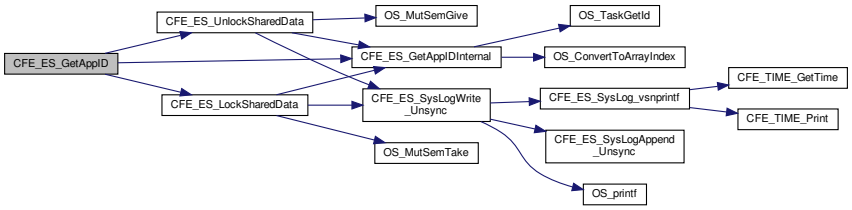
CFE_ES_GetResetType, CFE_ES_GetAppIDByName, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

Definition at line 801 of file cfe_es_api.c.

References CFE_ES_GetAppIDInternal(), CFE_ES_LockSharedData(), and CFE_ES_UnlockSharedData().

Referenced by CFE_ES_CDS_ValidateAppID(), CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_SB_↩ AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipe(), CFE_SB_GetLastSenderId(), CFE_SB_LockSharedData(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnlockSharedData(), CFE_↩ SB_Unsubscribe(), CFE_SB_UnsubscribeLocal(), CFE_SB_ZeroCopyGetPtr(), and EVS_GetAppID().

Here is the call graph for this function:



13.50.3.10 CFE_ES_GetAppIDByName()

```
int32 CFE_ES_GetAppIDByName (
            uint32 * AppIdPtr,
            const char * AppName )
```

Description

This routine retrieves the cFE Application ID associated with a specified Application name.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppIdPtr*	Pointer to variable that is to receive the Application's ID.
in	*AppName*	Pointer to null terminated character string containing an Application name.
out	∗*AppIdPtr*	Application ID of the calling Application.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

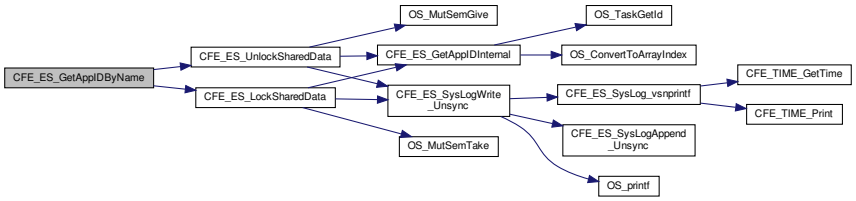
CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

Definition at line 765 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_APPNAME, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppStartParams_t::Name, OS_MAX_API_NAME, and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_DeleteCDS(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_RestartAppCmd(), CFE_ES_StopAppCmd(), and EVS_GetApplicationInfo().

Here is the call graph for this function:



13.50.3.11 CFE_ES_GetAppInfo()

```
int32 CFE_ES_GetAppInfo (
            CFE_ES_AppInfo_t * AppInfo,
            uint32 AppId )
```

Description

This routine retrieves the information about an App associated with a specified App ID. The information includes all of the information ES maintains for an application (documented in the CFE_ES_AppInfo_t type)

Assumptions, External Events, and Notes:

None

Parameters

in	*AppInfo*	Pointer to a CFE_ES_AppInfo_t structure that holds the specific Application information.
in	*AppId*	Application ID of Application whose name is being requested.
out	*AppInfo	Filled out CFE_ES_AppInfo_t structure containing the App Name, and application memory addresses among other fields.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

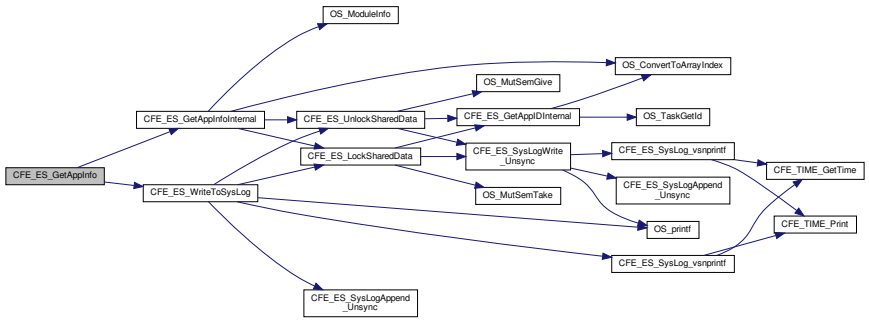
See also

CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName

Definition at line 872 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_APPID, CFE_ES_ERR_BUFFER, CFE_ES_GetAppInfoInternal(), CFE_ES_Global, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX_APPLICATIONS, and CFE_SUCCESS.

Here is the call graph for this function:



13.50.3.12 CFE_ES_GetAppName()

```
int32 CFE_ES_GetAppName (
            char * AppName,
            uint32 AppId,
            uint32 BufferLength )
```

Description

This routine retrieves the cFE Application name associated with a specified Application ID.

Assumptions, External Events, and Notes:

In the case of a failure (CFE_ES_ERR_APPID), an empty string is returned. CFE_ES_ERR_APPID will be returned if the specified Application ID (AppId) is invalid or not in use.

Parameters

in	*AppName*	Pointer to a character array of at least BufferLength in size that will be filled with the appropriate Application name.
in	*AppId*	Application ID of Application whose name is being requested.
in	*BufferLength*	The maximum number of characters, including the null terminator, that can be put into the AppName buffer. This routine will truncate the name to this length, if necessary.
out	∗*AppName*	Null terminated Application name of the Application associated with the specified Application ID.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.

Returns

See also

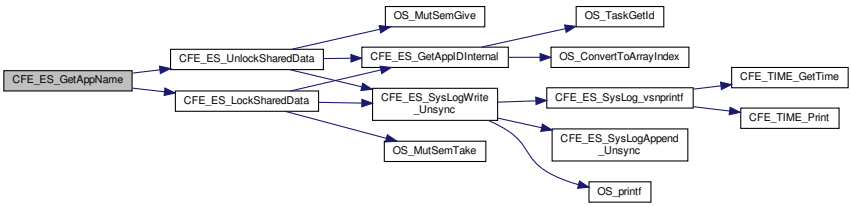
CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetTaskInfo

Definition at line 822 of file cfe_es_api.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppStartParams_t::Name, and CFE_ES_AppRecord_t::StartParams.

Referenced by CFE_ES_FormCDSName(), CFE_ES_RegisterCDS(), CFE_EVS_WriteAppDataFileCmd(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_SendRtgInfo(), CFE_SB_SetPipeOpts(), EVS_GenerateEventTelemetry(), EVS_IsFiltered(), and EVS_NotRegistered().

Here is the call graph for this function:



13.50.3.13 CFE_ES_GetGenCount()

```
int32 CFE_ES_GetGenCount (
            uint32 CounterId,
            uint32 * Count )
```

Description

This routine gets the value of a generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter to get the value from.
in	*Count	The value of the Counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_IncrementGenCounter, CFE_ES_GetGenCounterIDByName

Definition at line 1673 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, NULL, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.50.3.14 CFE_ES_GetGenCounterIDByName()

```
int32 CFE_ES_GetGenCounterIDByName (
            uint32 * CounterIdPtr,
            const char * CounterName )
```

Description

This routine gets the Counter Id for a generic counter specified by name.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterName	The name of the Counter.
out	*CounterIdPtr	The Counter Id for the given name.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_Increment↩GenCounter, CFE_ES_GetGenCount

Definition at line 1687 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_↩SUCCESS, CFE_ES_GenCounterRecord_t::CounterName, CFE_ES_Global_t::CounterTable, NULL, OS_MAX_API↩_NAME, and CFE_ES_GenCounterRecord_t::RecordUsed.

Referenced by CFE_ES_RegisterGenCounter().

13.50.3.15 CFE_ES_GetMemPoolStats()

```
int32 CFE_ES_GetMemPoolStats (
            CFE_ES_MemPoolStats_t * BufPtr,
            CFE_ES_MemHandle_t Handle )
```

Description

This routine fills the CFE_ES_MemPoolStats_t data structure with the statistics maintained by the memory pool software. These statistics can then be telemetered by the calling Application.

Assumptions, External Events, and Notes:

None

Parameters

in	*BufPtr*	Pointer to CFE_ES_MemPoolStats_t data structure to be filled with memory statistics.
in	*Handle*	The handle to the memory pool whose statistics are desired.
out	∗*BufPtr*	Memory Pool Statistics stored in given data structure.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.

Returns

See also

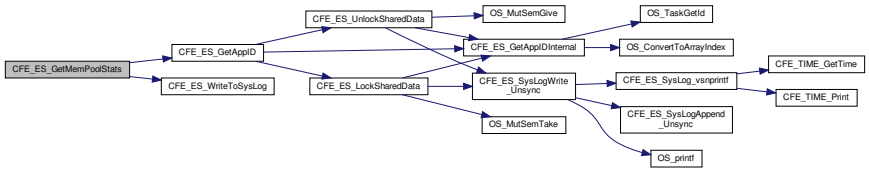
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf

Definition at line 627 of file cfe_esmempool.c.

References CFE_ES_BlockStats_t::BlockSize, CFE_ES_MemPoolStats_t::BlockStats, CFE_ES_ERR_MEM_HANDLE, CFE_ES_GetAppID(), CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_WriteToSysLog(), CFE_SUCCESS, Pool_t::CheckErrCntr, CFE_ES_MemPoolStats_t::CheckErrCtr, Pool_t::CurrentAddr, Pool_t::End, BlockSizeDesc_t::MaxSize, NULL, CFE_ES_MemPoolStats_t::NumBlocksRequested, BlockSizeDesc_t::NumCreated, CFE_ES_BlockStats_t::NumCreated, BlockSizeDesc_t::NumFree, CFE_ES_BlockStats_t::NumFree, CFE_ES_MemPoolStats_t::NumFreeBytes, Pool_t::PoolHandle, CFE_ES_MemPoolStats_t::PoolSize, Pool_t::RequestCntr, Pool_t::Size, and Pool_t::SizeDesc.

Referenced by CFE_ES_SendMemPoolStatsCmd().

Here is the call graph for this function:



13.50.3.16 CFE_ES_GetPoolBuf()

```
int32 CFE_ES_GetPoolBuf (
            uint32 ** BufPtr,
            CFE_ES_MemHandle_t HandlePtr,
            uint32 Size )
```

Description

This routine obtains a block of memory from the memory pool supplied by the calling application.

Assumptions, External Events, and Notes:

1. The size allocated from the memory pool is, at a minimum, 12 bytes more than requested.

Parameters

in	*BufPtr*	A pointer to the Application's pointer in which will be stored the address of the allocated memory buffer.
in	*HandlePtr*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*Size*	The size of the buffer requested. NOTE: The size allocated may be larger.
out	∗*BufPtr*	The address of the requested buffer.

When successful, the return value is a positive number and is the number of bytes actually allocated for the buffer.	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.
CFE_ES_ERR_MEM_BLOCK_SIZE	The block size requested is invalid.

Returns

See also

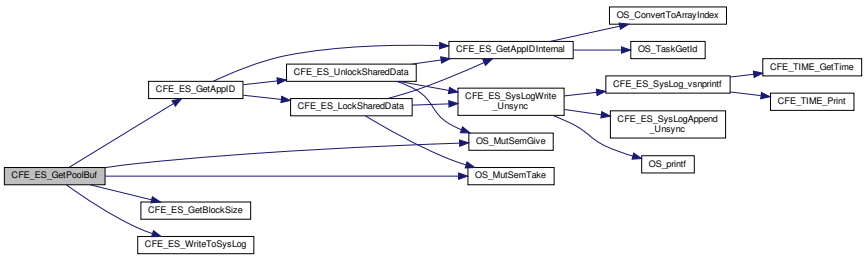
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_PutPoolBuf, CFE_ES_↩ GetMemPoolStats, CFE_ES_GetPoolBufInfo

Definition at line 307 of file cfe_esmempool.c.

References MemPoolAddr_t::Addr, Pool_t::AlignMask, BD::Allocated, MemPoolAddr_t::BdPtr, CFE_ES_CHECK_P↩ ATTERN, CFE_ES_ERR_MEM_BLOCK_SIZE, CFE_ES_ERR_MEM_HANDLE, CFE_ES_GetAppID(), CFE_ES_↩ GetBlockSize(), CFE_ES_MEMORY_ALLOCATED, CFE_ES_USE_MUTEX, CFE_ES_WriteToSysLog(), BD::Check↩ Bits, Pool_t::CurrentAddr, Pool_t::End, BlockSizeDesc_t::MaxSize, Pool_t::MutexId, BD::Next, NULL, BlockSizeDesc↩ _t::NumCreated, BlockSizeDesc_t::NumFree, OS_MutSemGive(), OS_MutSemTake(), Pool_t::PoolHandle, Pool_t::↩ RequestCntr, BD::Size, Pool_t::Size, Pool_t::SizeDesc, Pool_t::SizeDescPtr, BlockSizeDesc_t::Top, Pool_t::UseMutex, and MemPoolAddr_t::UserPtr.

Referenced by CFE_SB_AppInit(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), and CFE_SB_Zero↩ CopyGetPtr().

Here is the call graph for this function:



13.50.3.17 CFE_ES_GetPoolBufInfo()

```
int32 CFE_ES_GetPoolBufInfo (
            CFE_ES_MemHandle_t HandlePtr,
            uint32 * BufPtr )
```

Description

This routine gets info on a buffer in the memory pool.

Assumptions, External Events, and Notes:

None

Parameters

in	*HandlePtr*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*BufPtr*	A pointer to the memory buffer to provide status for.

When successful, the return value is a positive number and is the number of bytes actually allocated.	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.
CFE_ES_BUFFER_NOT_IN_POOL	The specified address is not in the memory pool.

Returns

See also

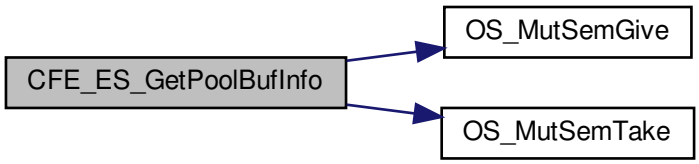
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_GetMemPoolStats, CFE_ES_PutPoolBuf

Definition at line 435 of file cfe_esmempool.c.

References MemPoolAddr_t::Addr, BD::Allocated, MemPoolAddr_t::BdPtr, CFE_ES_BUFFER_NOT_IN_POOL, CFE_ES_CHECK_PATTERN, CFE_ES_ERR_MEM_HANDLE, CFE_ES_MEMORY_ALLOCATED, CFE_ES_USE_MUTEX, BD::CheckBits, Pool_t::End, Pool_t::MutexId, NULL, OS_MutSemGive(), OS_MutSemTake(), Pool_t::PoolHandle, BD::Size, Pool_t::UseMutex, and MemPoolAddr_t::UserPtr.

Referenced by CFE_SB_ZeroCopyReleaseDesc().

Here is the call graph for this function:



13.50.3.18 CFE_ES_GetResetType()

```
int32 CFE_ES_GetResetType (
            uint32 * ResetSubtypePtr )
```

Description

Provides the caller with codes that identifies the type of Reset the processor most recently underwent. The caller can also obtain information on what caused the reset by supplying a pointer to a variable that will be filled with the Reset Sub-Type.

Assumptions, External Events, and Notes:

None

Parameters

in	*ResetSubtypePtr*	Pointer to uint32 type variable in which the Reset Sub-Type will be stored. The caller can set this pointer to NULL if the Sub-Type is of no interest.
out	**ResetSubtypePtr*	If the provided pointer was not NULL, the Reset Sub-Type is stored at the given address. For a list of possible Sub-Type values, see "Reset Sub-Types" .

CFE_PSP_RST_TYPE_POWERON	All memory has been cleared
CFE_PSP_RST_TYPE_PROCESSOR	Volatile disk, Critical Data Store and User Reserved memory could still be valid

Returns

See also

CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName, CFE_ES_GetTaskInfo

Definition at line 64 of file cfe_es_api.c.

References CFE_ES_ResetDataPtr, NULL, CFE_ES_ResetVariables_t::ResetSubtype, CFE_ES_ResetVariables_t::↩ ResetType, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_EVS_EarlyInit().

13.50.3.19 CFE_ES_GetTaskInfo()

```
int32 CFE_ES_GetTaskInfo (
            CFE_ES_TaskInfo_t * TaskInfo,
            uint32 TaskId )
```

Description

This routine retrieves the information about a Task associated with a specified Task ID. The information includes Task Name, and Parent/Creator Application ID.

Assumptions, External Events, and Notes:

None

Parameters

in	*TaskInfo*	Pointer to a CFE_ES_TaskInfo_t structure that holds the specific task information.
in	*TaskId*	Application ID of Application whose name is being requested.
out	∗*TaskInfo*	Filled out CFE_ES_TaskInfo_t structure containing the Task Name, Parent App Name, Parent App ID among other fields.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_TASKID	Occurs when the Task ID passed into CFE_ES_GetTaskInfo is invalid.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

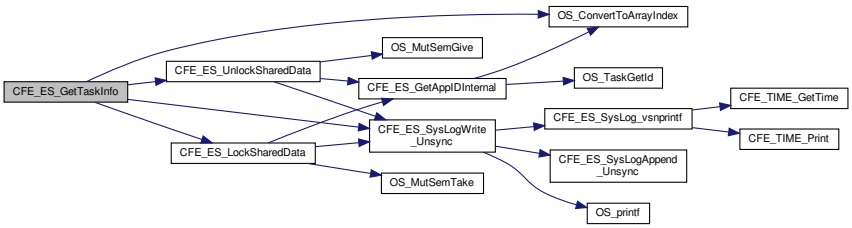
CFE_ES_GetResetType, CFE_ES_GetAppID, CFE_ES_GetAppIDByName, CFE_ES_GetAppName

Definition at line 912 of file cfe_es_api.c.

References CFE_ES_TaskRecord_t::AppId, CFE_ES_TaskInfo_t::AppId, CFE_ES_TaskInfo_t::AppName, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_UNDEFINED, CFE_ES_ERR_TASKID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_SUCCESS, CFE_ES_TaskRecord_t::ExecutionCounter, CFE_ES_TaskInfo_t::ExecutionCounter, CFE_ES_AppStartParams_t::Name, OS_ConvertToArrayIndex(), OS_MAX_API_NAME, OS_MAX_TASKS, OS_SUCCESS, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskInfo_t::TaskId, CFE_ES_TaskRecord_t::TaskName, CFE_ES_TaskInfo_t::TaskName, and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_ListTasks(), CFE_ES_ProcessCoreException(), CFE_ES_QueryAllTasksCmd(), and CFE_SB_GetAppTskName().

Here is the call graph for this function:



13.50.3.20 CFE_ES_IncrementGenCounter()

```
int32 CFE_ES_IncrementGenCounter (
            uint32 CounterId )
```

Description

This routine increments the specified generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter to be incremented.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

Definition at line 1634 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_↩ SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, and CFE_ES_GenCounter↩ Record_t::RecordUsed.

13.50.3.21 CFE_ES_IncrementTaskCounter()

```
void CFE_ES_IncrementTaskCounter (
            void  )
```

Description

This routine increments the execution counter that is stored for the calling task. It can be called from cFE Application main tasks, child tasks, or cFE Core application main tasks. Normally, the call is not necessary from a cFE Application, since the CFE_ES_RunLoop call increments the counter for the Application.

Assumptions, External Events, and Notes:

NOTE: This API is not needed for Appplications that call the CFE_ES_RunLoop call.

This function does not return a value.

Returns

See also

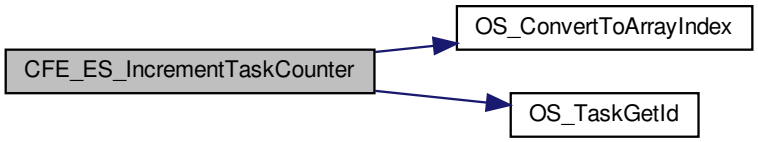
CFE_ES_RunLoop

Definition at line 1153 of file cfe_es_api.c.

References CFE_ES_Global, CFE_ES_TaskRecord_t::ExecutionCounter, OS_ConvertToArrayIndex(), OS_SUCCESS, OS_TaskGetId(), and CFE_ES_Global_t::TaskTable.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.50.3.22 CFE_ES_Main()

```
void CFE_ES_Main (
             uint32 StartType,
             uint32 StartSubtype,
             uint32 ModeId,
             const char * StartFilePath )
```

Description

cFE main entry point. This is the entry point into the cFE software. It is called only by the Board Support Package software.

Assumptions, External Events, and Notes:

None

Parameters

in	*StartType*	Identifies whether this was a CFE_PSP_RST_TYPE_POWERON or CFE_PSP_RST_TYPE_PROCESSOR.
in	*StartSubtype*	Specifies, in more detail, what caused the `StartType` identified above. See CFE_PSP_RST_SUBTYPE_POWER_CYCLE for possible examples.
in	*ModeId*	Identifies the source of the Boot as determined by the BSP.
in	*StartFilePath*	Identifies the startup file to use to initialize the cFE apps.

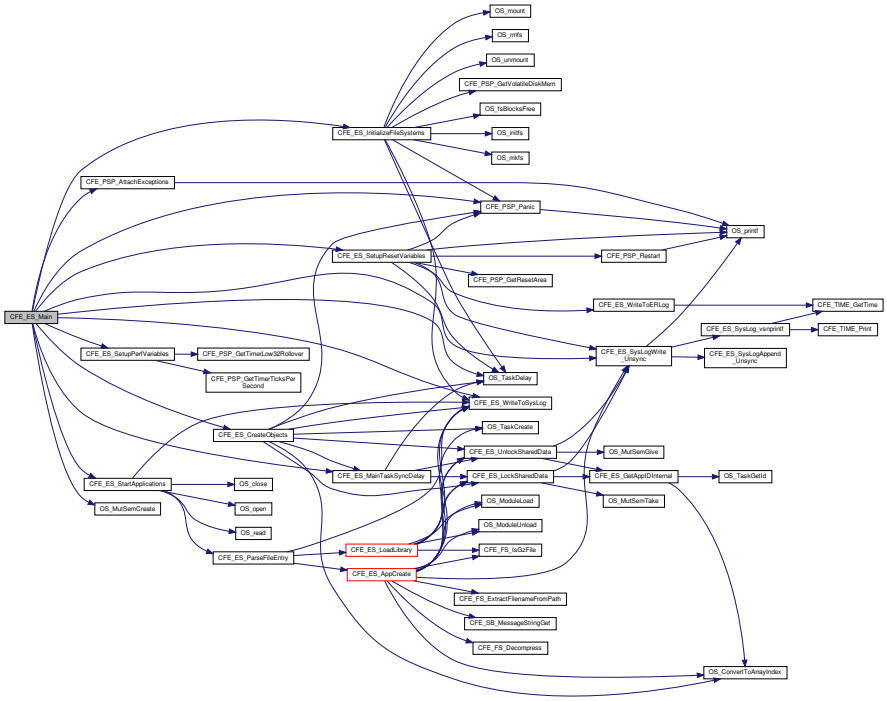
See also

CFE_ES_ResetCFE

Definition at line 86 of file cfe_es_start.c.

References CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_AppState_LATE_INIT, CFE_ES_AppState_RUNNING, CFE_ES_AppState_UNDEFINED, CFE_ES_CreateObjects(), CFE_ES_InitializeFileSystems(), CFE_ES_MainTaskSyncDelay(), CFE_ES_PANIC_DELAY, CFE_ES_SetupPerfVariables(), CFE_ES_SetupResetVariables(), CFE_ES_StartApplications(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_SystemState_APPS_INIT, CFE_ES_SystemState_CORE_READY, CFE_ES_SystemState_CORE_STARTUP, CFE_ES_SystemState_EARLY_INIT, CFE_ES_SystemState_OPERATIONAL, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC, CFE_PSP_AttachExceptions(), CFE_PSP_Panic(), CFE_PSP_PANIC_STARTUP_SEM, CFE_SUCCESS, CFE_ES_Global_t::CounterTable, OS_MAX_TASKS, OS_MutSemCreate(), OS_SUCCESS, OS_TaskDelay(), CFE_ES_GenCounterRecord_t::RecordUsed, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Global_t::SharedDataMutex, CFE_ES_Global_t::SystemState, and CFE_ES_Global_t::TaskTable.

Here is the call graph for this function:



13.50.3.23 CFE_ES_PerfLogAdd()

```
void CFE_ES_PerfLogAdd (
            uint32 Marker,
            uint32 EntryExit )
```

Description

This function logs the entry and exit marker for the specified `id`. This function is used by the Software Performance Analysis tool (see section 5.15).

Assumptions, External Events, and Notes:

None

Parameters

in	*Marker*	Identifier of the specific event or marker.
in	*EntryExit*	Used to specify Entry(0) or Exit(1)

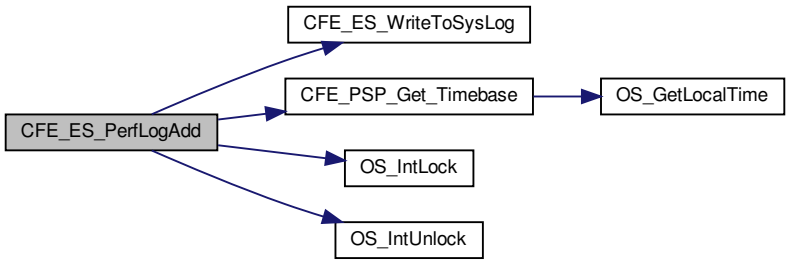
See also

CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit

Definition at line 415 of file cfe_es_perf.c.

References CFE_ES_PERF_IDLE, CFE_ES_PERF_TRIGGER_CENTER, CFE_ES_PERF_TRIGGER_END, CFE_ES_PERF_TRIGGER_START, CFE_ES_PERF_TRIGGERED, CFE_ES_PERF_WAITING_FOR_TRIGGER, CFE_ES_TEST_LONG_MASK, CFE_ES_WriteToSysLog(), CFE_MISSION_ES_PERF_EXIT_BIT, CFE_MISSION_ES_PERF_MAX_IDS, CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE, CFE_PSP_Get_Timebase(), CFE_ES_PerfDataEntry_t::Data, CFE_ES_PerfData_t::DataBuffer, CFE_ES_PerfMetaData_t::DataCount, CFE_ES_PerfMetaData_t::DataEnd, CFE_ES_PerfMetaData_t::DataStart, CFE_ES_PerfMetaData_t::FilterMask, CFE_ES_PerfMetaData_t::InvalidMarkerReported, CFE_ES_PerfData_t::MetaData, CFE_ES_PerfMetaData_t::Mode, OS_IntLock(), OS_IntUnlock(), CFE_ES_PerfMetaData_t::State, CFE_ES_PerfDataEntry_t::TimerLower32, CFE_ES_PerfDataEntry_t::TimerUpper32, CFE_ES_PerfMetaData_t::TriggerCount, and CFE_ES_PerfMetaData_t::TriggerMask.

Here is the call graph for this function:



13.50.3.24 CFE_ES_PoolCreate()

```
int32 CFE_ES_PoolCreate (
            CFE_ES_MemHandle_t * HandlePtr,
            uint8 * MemPtr,
            uint32 Size )
```

Description

This routine initializes a pool of memory supplied by the calling application. When a memory pool created by this routine is processed, mutex handling will be performed.

Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

Parameters

in	*HandlePtr*	A pointer to the variable the caller wishes to have the memory pool handle kept in.
in	*MemPtr*	A Pointer to the pool of memory created by the calling application. This address must be on a 32-bit boundary.
in	*Size*	The size of the pool of memory. Note that this must be an integral number of 32 bit words.
out	∗*HandlePtr*	The memory pool handle.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

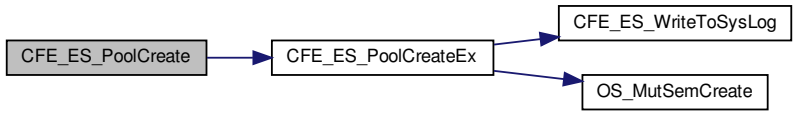
See also

CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf, CFE_ES_GetMemPoolStats

Definition at line 129 of file cfe_esmempool.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_MemPoolDefSize, CFE_ES_PoolCreateEx(), and CFE_ES_USE_MUTEX.

Here is the call graph for this function:



13.50.3.25 CFE_ES_PoolCreateEx()

```
int32 CFE_ES_PoolCreateEx (
            CFE_ES_MemHandle_t * HandlePtr,
            uint8 * MemPtr,
            uint32 Size,
            uint32 NumBlockSizes,
            uint32 * BlockSizes,
            uint16 UseMutex )
```

Description

This routine initializes a pool of memory supplied by the calling application.

Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

Parameters

in	*HandlePtr*	A pointer to the variable the caller wishes to have the memory pool handle kept in.
in	*MemPtr*	A Pointer to the pool of memory created by the calling application. This address must be on a 32-bit boundary.
in	*Size*	The size of the pool of memory. Note that this must be an integral number of 32 bit words.
in	*NumBlockSizes*	The number of different block sizes specified in the `BlockSizes` array. If set equal to zero or if greater than 17, then default block sizes are used.
in	*BlockSizes*	Pointer to an array of sizes to be used instead of the default block sizes specified by CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 through CFE_PLATFORM_ES_MAX_BLOCK_SIZE. If the pointer is equal to NULL, the default block sizes are used.
in	*UseMutex*	Flag indicating whether the new memory pool will be processing with mutex handling or not. Valid parameter values are CFE_ES_USE_MUTEX and CFE_ES_NO_MUTEX
out	∗*HandlePtr*	The memory pool handle.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

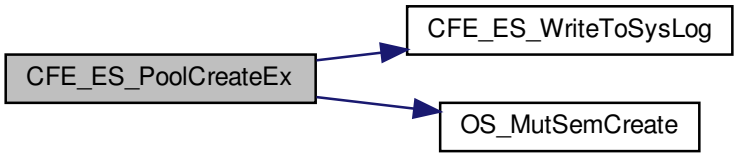
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf, CFE_ES_↩ GetMemPoolStats

Definition at line 138 of file cfe_esmempool.c.

References ALIGN_OF, Pool_t::AlignMask, CFE_ES_BAD_ARGUMENT, CFE_ES_MAX_MEMPOOL_BLOCK_SI↩ ZES, CFE_ES_MemPoolDefSize, CFE_ES_NO_MUTEX, CFE_ES_STATIC_POOL_TYPE, CFE_ES_USE_MUTEX, CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_MEMPOOL_ALIGN_SIZE_MIN, CFE_SUCCESS, Pool_t::Check↩ ErrCntr, Pool_t::CurrentAddr, Pool_t::End, BlockSizeDesc_t::MaxSize, Pool_t::MutexId, NULL, BlockSizeDesc_t::↩ NumCreated, BlockSizeDesc_t::NumFree, OS_MAX_API_NAME, OS_MutSemCreate(), Pool_t::PoolHandle, Pool_↩ t::RequestCntr, Pool_t::Size, Pool_t::SizeDesc, Pool_t::SizeDescPtr, BlockSizeDesc_t::Top, and Pool_t::UseMutex.

Referenced by CFE_ES_PoolCreate(), CFE_ES_PoolCreateNoSem(), and CFE_SB_InitBuffers().

Here is the call graph for this function:



13.50.3.26 CFE_ES_PoolCreateNoSem()

```
int32 CFE_ES_PoolCreateNoSem (
            CFE_ES_MemHandle_t * HandlePtr,
            uint8 * MemPtr,
            uint32 Size )
```

Description

This routine initializes a pool of memory supplied by the calling application. When a memory pool created by this routine is processed, no mutex handling is performed.

Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

Parameters

in	*HandlePtr*	A pointer to the variable the caller wishes to have the memory pool handle kept in.
in	*MemPtr*	A Pointer to the pool of memory created by the calling application. This address must be on a 32-bit boundary.
in	*Size*	The size of the pool of memory. Note that this must be an integral number of 32 bit words.
out	∗*HandlePtr*	The memory pool handle.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

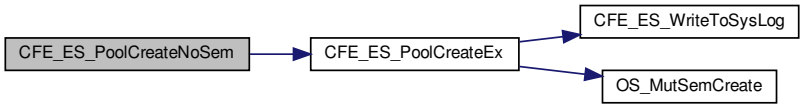
See also

CFE_ES_PoolCreate, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_PutPoolBuf, CFE_ES_GetMemPoolStats

Definition at line 118 of file cfe_esmempool.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_MemPoolDefSize, CFE_ES_NO_MUTEX, and CFE_ES_PoolCreateEx().

Here is the call graph for this function:



13.50.3.27 CFE_ES_ProcessCoreException()

```
void CFE_ES_ProcessCoreException (
            uint32 HostTaskId,
            const char * ReasonString,
            const uint32 * ContextPointer,
            uint32 ContextSize )
```

Description

This hook routine is called from the PSP when an exception occurs

Assumptions, External Events, and Notes:

None.

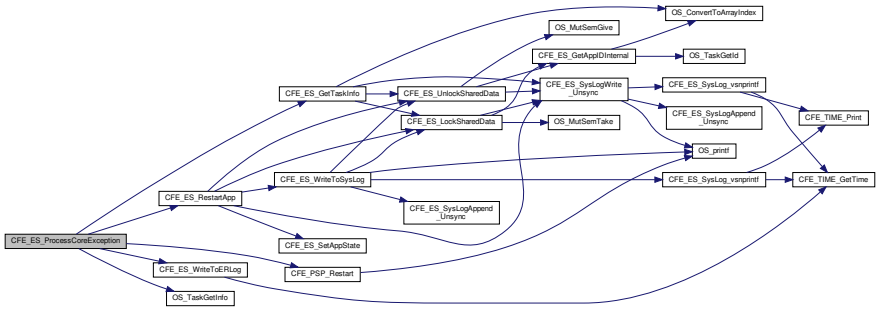
Parameters

in	*HostTaskId*	The OS (not OSAL) task ID
in	*ReasonString*	Identifier from PSP
in	*ContextPointer*	Context data from PSP
in	*ContextSize*	Size of context data from PSP

Definition at line 1852 of file cfe_es_api.c.

References CFE_ES_TaskInfo_t::AppId, CFE_ES_Global_t::AppTable, CFE_ES_APP_RESTART, CFE_ES_↩ ExceptionAction_RESTART_APP, CFE_ES_GetTaskInfo(), CFE_ES_Global, CFE_ES_LogEntryType_CORE, CF↩ E_ES_ResetDataPtr, CFE_ES_RestartApp(), CFE_ES_WriteToERLog(), CFE_PSP_Restart(), CFE_PSP_RST_SU↩ BTYPE_EXCEPTION, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_SUCCE↩ SS, CFE_ES_ResetVariables_t::ES_CausedReset, CFE_ES_AppStartParams_t::ExceptionAction, CFE_ES_Reset↩ Variables_t::MaxProcessorResetCount, OS_MAX_TASKS, OS_SUCCESS, OS_TaskGetInfo(), OS_task_prop_t::O↩ Stask_id, CFE_ES_ResetVariables_t::ProcessorResetCount, CFE_ES_TaskRecord_t::RecordUsed, CFE_ES_Reset↩ Data_t::ResetVars, CFE_ES_AppRecord_t::StartParams, CFE_ES_TaskRecord_t::TaskId, and CFE_ES_Global_t::↩ TaskTable.

Here is the call graph for this function:



13.50.3.28 CFE_ES_PutPoolBuf()

```
int32 CFE_ES_PutPoolBuf (
            CFE_ES_MemHandle_t HandlePtr,
            uint32 * BufPtr )
```

Description

This routine releases a buffer back into the memory pool.

Assumptions, External Events, and Notes:

None

Parameters

in	*HandlePtr*	The handle to the memory pool as returned by CFE_ES_PoolCreate or CFE_ES_PoolCreateNoSem.
in	*BufPtr*	A pointer to the memory buffer to be released.

When successful, the return value is a positive number and is the number of bytes actually released.	
CFE_ES_ERR_MEM_HANDLE	The Memory Pool handle is invalid.

Returns

See also

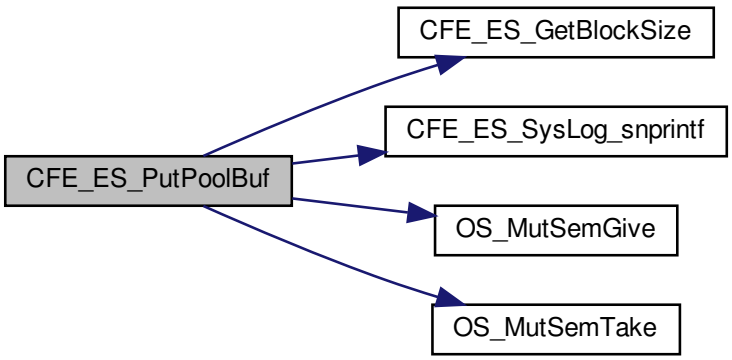
CFE_ES_PoolCreate, CFE_ES_PoolCreateNoSem, CFE_ES_PoolCreateEx, CFE_ES_GetPoolBuf, CFE_ES_↩ GetMemPoolStats, CFE_ES_GetPoolBufInfo

Definition at line 492 of file cfe_esmempool.c.

References MemPoolAddr_t::Addr, BD::Allocated, MemPoolAddr_t::BdPtr, CFE_ES_CHECK_PATTERN, CFE_ES_↩ ERR_MEM_HANDLE, CFE_ES_GetBlockSize(), CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_MEMORY_ALLO↩ CATED, CFE_ES_MEMORY_DEALLOCATED, CFE_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_ES↩ _USE_MUTEX, BD::CheckBits, Pool_t::CheckErrCntr, Pool_t::End, BlockSizeDesc_t::MaxSize, Pool_t::MutexId, BD::↩ Next, NULL, BlockSizeDesc_t::NumFree, OS_MutSemGive(), OS_MutSemTake(), Pool_t::PoolHandle, BD::Size, Pool↩ _t::SizeDesc, Pool_t::SizeDescPtr, BlockSizeDesc_t::Top, Pool_t::UseMutex, and MemPoolAddr_t::UserPtr.

Referenced by CFE_SB_AppInit(), CFE_SB_PutDestinationBlk(), CFE_SB_ReturnBufferToPool(), CFE_SB_Zero↩ CopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

Here is the call graph for this function:



13.50.3.29 CFE_ES_RegisterApp()

```
int32 CFE_ES_RegisterApp (
            void  )
```

Description

This API registers the calling Application with the cFE.

Assumptions, External Events, and Notes:

NOTE: This function **MUST** be called before any other cFE API functions are called.

Return codes from OS_TaskRegister	
Return codes from OS_BinSemTake	
CFE_SUCCESS	Operation was performed successfully

Returns

See also

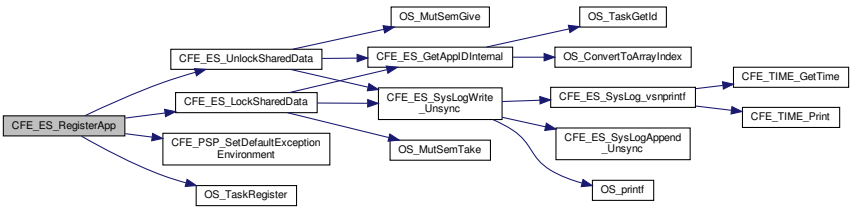
CFE_ES_ExitApp, CFE_ES_RunLoop

Definition at line 722 of file cfe_es_api.c.

References CFE_ES_ERR_APP_REGISTER, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_P↩SP_SetDefaultExceptionEnvironment(), CFE_SUCCESS, OS_SUCCESS, and OS_TaskRegister().

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), and CFE_SB_AppInit().

Here is the call graph for this function:



13.50.3.30 CFE_ES_RegisterCDS()

```
int32 CFE_ES_RegisterCDS (
            CFE_ES_CDSHandle_t * HandlePtr,
            int32 BlockSize,
            const char * Name )
```

Description

This routine allocates a block of memory in the Critical Data Store and associates it with the calling Application. The memory can survive an Application restart as well as a Processor Reset.

Assumptions, External Events, and Notes:

None

Parameters

in	*HandlePtr*	Pointer Application's variable that will contain the CDS Memory Block Handle.
in	*BlockSize*	The number of bytes needed in the CDS.
in	*Name*	A pointer to a character string containing an application unique name of CFE_MISSION_ES_CDS_MAX_NAME_LENGTH characters or less.
out	*∗HandlePtr*	The handle of the CDS block that can be used in CFE_ES_CopyToCDS and CFE_ES_RestoreFromCDS.

CFE_SUCCESS	The memory block was successfully created in the CDS.
CFE_ES_NOT_IMPLEMENTED	The processor does not support a Critical Data Store.
CFE_ES_CDS_ALREADY_EXISTS	The Application is receiving the pointer to a CDS that was already present.
CFE_ES_CDS_INVALID_SIZE	The Application is requesting a CDS Block with a size of zero.
CFE_ES_CDS_INVALID_NAME	The Application is requesting a CDS Block with an invalid ASCII string name. Either the name is too long (> CFE_MISSION_ES_CDS_MA↩ X_NAME_LENGTH) or was an empty string.
CFE_ES_CDS_REGISTRY_FULL	The CDS Registry has as many entries in it as it can hold. The CDS Registry size can be adjusted with the CFE_PLATFORM_ES_CDS_↩ MAX_NUM_ENTRIES macro defined in the cfe_platform_cfg.h file.

Returns

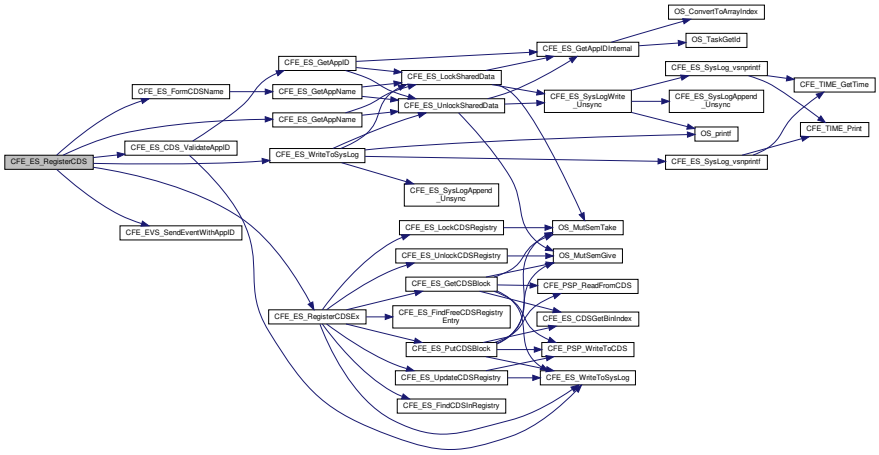
See also

CFE_ES_CopyToCDS, CFE_ES_RestoreFromCDS

Definition at line 1466 of file cfe_es_api.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_BAD_HANDLE, CFE_ES_CDS_INVALID_NAME, CFE_ES_CDS_INVALID_SIZE, CFE_ES_CDS_MAX_FULL_NAME_LEN, CFE_ES_CDS_REGISTER_ERR_EID, CFE_ES_CDS_ValidateAppID(), CFE_ES_FormCDSName(), CFE_ES_GetAppName(), CFE_ES_Global, CFE_ES_NOT_IMPLEMENTED, CFE_ES_RegisterCDSEx(), CFE_ES_WriteToSysLog(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_MISSION_ES_CDS_MAX_NAME_LENGTH, CFE_SUCCESS, CFE_ES_CDS_Variables_t::MemPoolSize, and OS_MAX_API_NAME.

Here is the call graph for this function:



13.50.3.31 CFE_ES_RegisterChildTask()

```
int32 CFE_ES_RegisterChildTask (
            void  )
```

Description

This routine registers a cFE Child task and associates it with its parent cFE Application.

Assumptions, External Events, and Notes:

NOTE: This API **MUST** be called by the Child Task before any other cFE API calls are made.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_CHILD_TASK_REGISTER	Errors occured when trying to register a child task.

Returns

See also

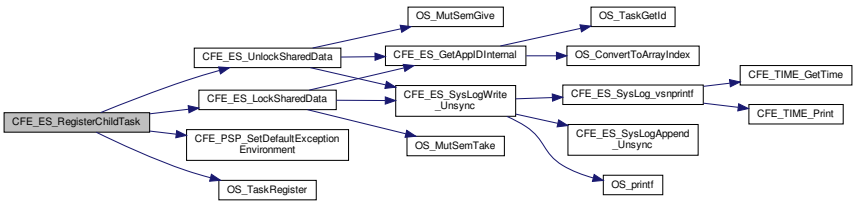
CFE_ES_CreateChildTask, CFE_ES_DeleteChildTask, CFE_ES_ExitChildTask

Definition at line 1111 of file cfe_es_api.c.

References CFE_ES_ERR_CHILD_TASK_REGISTER, CFE_ES_LockSharedData(), CFE_ES_UnlockSharedData(), CFE_PSP_SetDefaultExceptionEnvironment(), CFE_SUCCESS, OS_SUCCESS, and OS_TaskRegister().

Referenced by CFE_ES_PerfLogDump().

Here is the call graph for this function:



13.50.3.32 CFE_ES_RegisterGenCounter()

```
int32 CFE_ES_RegisterGenCounter (
            uint32 * CounterIdPtr,
            const char * CounterName )
```

Description

This routine registers a generic counter.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterName	The Name of the generic counter.
out	*CounterIdPtr	The Counter Id of the newly created counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

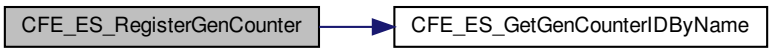
See also

CFE_ES_IncrementGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_SetGenCount, CFE_ES_GetGen↩ Count, CFE_ES_GetGenCounterIDByName

Definition at line 1573 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_GetGenCounterIDByName(), CFE_ES_Global, CFE_PLAT↩ FORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Gen↩ CounterRecord_t::CounterName, CFE_ES_Global_t::CounterTable, NULL, OS_MAX_API_NAME, and CFE_ES_↩ GenCounterRecord_t::RecordUsed.

Here is the call graph for this function:



13.50.3.33 CFE_ES_ReloadApp()

```
int32 CFE_ES_ReloadApp (
            uint32 AppID,
            const char * AppFileName )
```

Description

This API causes a cFE Application to be stopped and restarted from the specified file.

Assumptions, External Events, and Notes:

The specified application will be deleted before it is reloaded from the specified file. In the event that an application cannot be reloaded due to a corrupt file, the application may no longer be reloaded when given a valid load file (it has been deleted and no longer exists). To recover, the application may be restarted by loading the application via the ES_STARTAPP command (CFE_ES_START_APP_CC).

Parameters

in	*AppID*	Identifies the application to be reset.
in	*AppFileName*	Identifies the new file to start.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

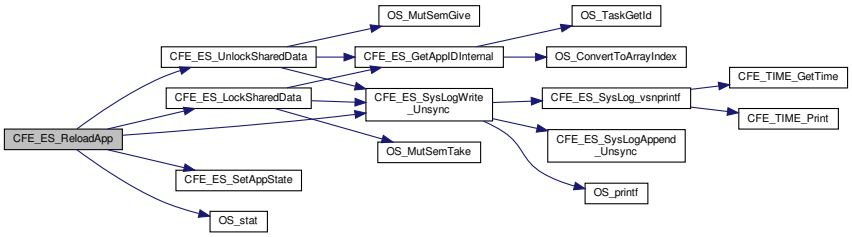
CFE_ES_RestartApp, CFE_ES_DeleteApp, CFE_ES_START_APP_CC

Definition at line 276 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_RUNNING, CFE_ES_AppState_WAITING, CFE_ES_AppType_CORE, CFE_ES_ERR_APPID, CFE_ES_FILE_IO_ERR, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_SYS_RELOAD, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_APP_KILL_TIMEOUT, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::FileName, CFE_ES_AppStartParams_t::Name, OS_MAX_PATH_LEN, OS_stat(), OS_SUCCESS, CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ReloadAppCmd().

Here is the call graph for this function:



13.50.3.34 CFE_ES_ResetCFE()

```
int32 CFE_ES_ResetCFE (
            uint32 ResetType )
```

Description

This API causes an immediate reset of the cFE Kernel and all cFE Applications. The caller can specify whether the reset should clear all memory (CFE_PSP_RST_TYPE_POWERON) or try to retain volatile memory areas (CFE_PSP_RST_TYPE_PROCESSOR).

Assumptions, External Events, and Notes:

None

Parameters

in	*ResetType*	Identifies the type of reset desired. Allowable settings are: • CFE_PSP_RST_TYPE_POWERON - Causes all memory to be cleared • CFE_PSP_RST_TYPE_PROCESSOR - Attempts to retain volatile disk, critical data store and user reserved memory.

CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.
CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

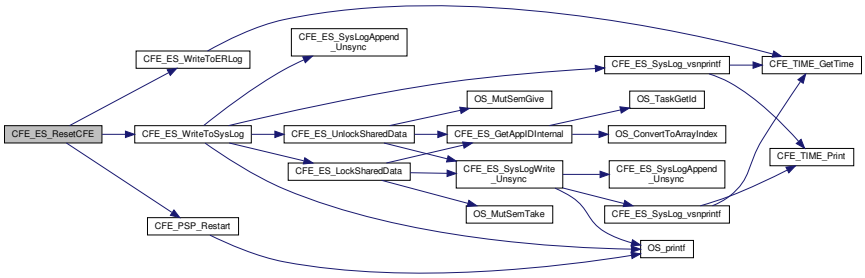
CFE_ES_Main

Definition at line 82 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_LogEntryType_CORE, CFE_ES_NOT_IMPLEMENTED, CFE_ES_ResetDataPtr, CFE_ES_WriteToERLog(), CFE_ES_WriteToSysLog(), CFE_PSP_Restart(), CFE_PSP_RST_SUBTYPE_RESET_COMMAND, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, CFE_ES_ResetVariables_t::ES_CausedReset, CFE_ES_ResetVariables_t::MaxProcessorResetCount, NULL, CFE_ES_ResetVariables_t::ProcessorResetCount, and CFE_ES_ResetData_t::ResetVars.

Referenced by CFE_ES_ExitApp(), and CFE_ES_RestartCmd().

Here is the call graph for this function:



13.50.3.35 CFE_ES_RestartApp()

```
int32 CFE_ES_RestartApp (
            uint32 AppID )
```

Description

This API causes a cFE Application to be stopped and restarted.

Assumptions, External Events, and Notes:

None

Parameters

in	*AppID*	Identifies the application to be reset.

CFE_ES_NOT_IMPLEMENTED	Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Returns

See also

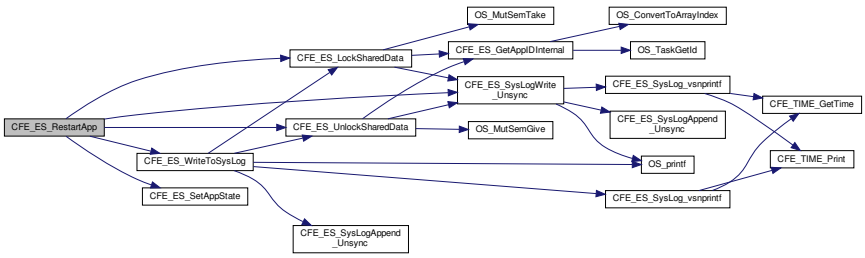
CFE_ES_ReloadApp, CFE_ES_DeleteApp

Definition at line 222 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_AppRecord_t::AppState, CFE_ES_Global_t::AppTable, CFE_ES_ControlReq_t::AppTimer, CFE_ES_AppState_RUNNING, CFE_ES_AppState_WAITING, CFE_ES_AppType_CORE, CFE_ES_ERR_APPID, CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_SYS_RESTART, CFE_ES_SetAppState(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_UnlockSharedData(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_ES_APP_KILL_TIMEOUT, CFE_PLATFORM_ES_MAX_APPLICATIONS, CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_AppStartParams_t::Name, CFE_ES_AppRecord_t::StartParams, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_ProcessCoreException(), and CFE_ES_RestartAppCmd().

Here is the call graph for this function:



13.50.3.36 CFE_ES_RestoreFromCDS()

```
int32 CFE_ES_RestoreFromCDS (
            void * RestoreToMemory,
            CFE_ES_CDSHandle_t Handle )
```

Description

This routine copies data from the Critical Data Store identified with the `Handle` into the area of memory pointed to by the `RestoreToMemory` pointer. The area of memory to be copied into must be at least as big as the size specified when registering the CDS. The recovery will indicate an error if the data integrity check maintained by the CDS indicates the contents of the CDS have changed. However, the contents will still be copied into the specified area of memory.

Assumptions, External Events, and Notes:

None

Parameters

in	*Handle*	The handle of the CDS block that was previously obtained from CFE_ES_RegisterCDS.
in	*RestoreToMemory*	A Pointer to the block of memory that is to be restored with the contents of the CDS.
out	∗*RestoreToMemory*	The contents of the specified CDS.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_CDS_BLOCK_CRC_ERR	Occurs when trying to read a CDS Data block and the CRC of the current data does not match the stored CRC for the data. Either the contents of the CDS Data Block are corrupted or the CDS Control Block is corrupted.
OS_ERROR	Problem with handle or a size mismatch

Returns

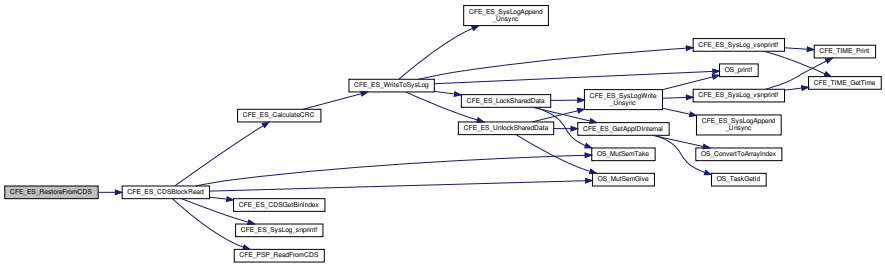
See also

CFE_ES_RegisterCDS, CFE_ES_CopyToCDS

Definition at line 1561 of file cfe_es_api.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDSBlockRead(), CFE_ES_Global, CFE_ES_CDS_RegRec_t::↩ MemHandle, and CFE_ES_CDSVariables_t::Registry.

Here is the call graph for this function:



13.50.3.37 CFE_ES_RunLoop()

```
bool CFE_ES_RunLoop (
            uint32 * ExitStatus )
```

Description

This is the API that allows an app to check for exit requests from the system.

Assumptions, External Events, and Notes:

None

Parameters

in	*ExitStatus*	A pointer to a variable containing the Application's desired run status. Acceptable values are: • CFE_ES_RunStatus_APP_RUN - Indicates that the Application should continue to run. • CFE_ES_RunStatus_APP_EXIT - Indicates that the Application wants to exit normally. • CFE_ES_RunStatus_APP_ERROR - Indicates that the Application is quitting with an error.

true	The application should continue executing
false	The application should terminate itself

Returns

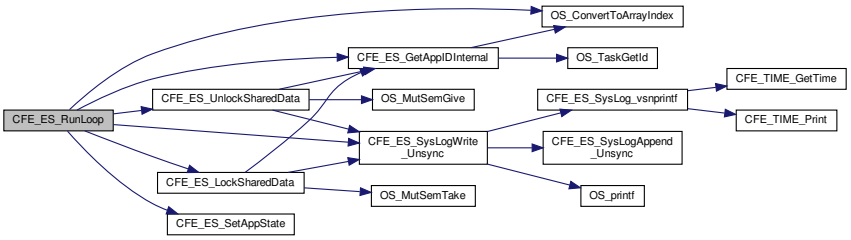
See also

CFE_ES_ExitApp, CFE_ES_RegisterApp

Definition at line 503 of file cfe_es_api.c.

References CFE_ES_ControlReq_t::AppControlRequest, CFE_ES_Global_t::AppTable, CFE_ES_AppState_RUNNI↩
NG, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_RunStatus_APP_ERR↩
OR, CFE_ES_RunStatus_APP_EXIT, CFE_ES_RunStatus_APP_RUN, CFE_ES_SetAppState(), CFE_ES_SysLog↩
Write_Unsync(), CFE_ES_UnlockSharedData(), CFE_SUCCESS, CFE_ES_AppRecord_t::ControlReq, CFE_ES_↩
TaskRecord_t::ExecutionCounter, CFE_ES_MainTaskInfo_t::MainTaskId, OS_ConvertToArrayIndex(), CFE_ES_App↩
Record_t::TaskInfo, and CFE_ES_Global_t::TaskTable.

Here is the call graph for this function:



13.50.3.38 CFE_ES_SetGenCount()

```
int32 CFE_ES_SetGenCount (
            uint32 CounterId,
            uint32 Count )
```

Description

This routine sets the specified generic counter to the specified value.

Assumptions, External Events, and Notes:

None.

Parameters

in	*CounterId*	The Counter to be set.
in	*Count*	The new value of the Counter.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_BAD_ARGUMENT	Bad parameter passed into an ES API.

Returns

See also

CFE_ES_RegisterGenCounter, CFE_ES_DeleteGenCounter, CFE_ES_IncrementGenCounter, CFE_ES_GetGenCount, CFE_ES_GetGenCounterIDByName

Definition at line 1654 of file cfe_es_api.c.

References CFE_ES_BAD_ARGUMENT, CFE_ES_Global, CFE_PLATFORM_ES_MAX_GEN_COUNTERS, CFE_SUCCESS, CFE_ES_GenCounterRecord_t::Counter, CFE_ES_Global_t::CounterTable, and CFE_ES_GenCounterRecord_t::RecordUsed.

13.50.3.39 CFE_ES_WaitForStartupSync()

```
void CFE_ES_WaitForStartupSync (
            uint32 TimeOutMilliseconds )
```

Description

This is the API that allows an app to wait for the rest of the apps to complete their entire initialization before continuing. It is most useful for applications such as Health and Safety or the Scheduler that need to wait until applications exist and are running before sending out packets to them.

This is a specialized wrapper for CFE_ES_WaitForSystemState() for compatibility with applications using this API.

Assumptions, External Events, and Notes:

This API should only be called as the last item of an Apps initialization. In addition, this API should only be called by an App that is started from the ES Startup file. It should not be used by an App that is started after the system is running. (Although it will cause no harm)

Parameters

in	*TimeOutMilliseconds*	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.

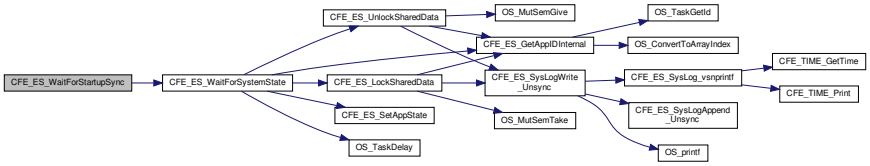
See also

CFE_ES_RunLoop

Definition at line 709 of file cfe_es_api.c.

References CFE_ES_SystemState_OPERATIONAL, and CFE_ES_WaitForSystemState().

Here is the call graph for this function:



13.50.3.40 CFE_ES_WaitForSystemState()

```
int32 CFE_ES_WaitForSystemState (
            uint32 MinSystemState,
            uint32 TimeOutMilliseconds )
```

Description

This is the API that allows an app to wait for the rest of the apps to complete a given stage of initialization before continuing.

This gives finer grained control than the "CFE_ES_WaitForStartupSync()" call.

Assumptions, External Events, and Notes:

This API assumes that the caller has also been initialized sufficiently to satisfy the global system state it is waiting for, and the apps own state will be updated accordingly.

Parameters

in	*TimeOutMilliseconds*	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.
in	*MinSystemState*	Determine the state of the App

Returns

if state was successfully achieved CFE_ES_OPERATION_TIMED_OUT if the timeout was reached (or other defined error code in case of error)

See also

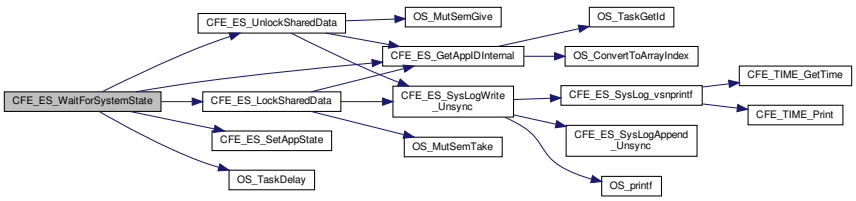
CFE_ES_RunLoop

Definition at line 607 of file cfe_es_api.c.

References CFE_ES_Global_t::AppTable, CFE_ES_AppState_EARLY_INIT, CFE_ES_AppState_LATE_INIT, CFE_ES_AppState_RUNNING, CFE_ES_AppState_STOPPED, CFE_ES_AppType_CORE, CFE_ES_GetAppIDInternal(), CFE_ES_Global, CFE_ES_LockSharedData(), CFE_ES_OPERATION_TIMED_OUT, CFE_ES_SetAppState(), CFE_ES_SystemState_APPS_INIT, CFE_ES_SystemState_CORE_READY, CFE_ES_SystemState_OPERATIONAL, CFE_ES_SystemState_SHUTDOWN, CFE_ES_UnlockSharedData(), CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC, CFE_SUCCESS, OS_TaskDelay(), CFE_ES_Global_t::SystemState, and CFE_ES_AppRecord_t::Type.

Referenced by CFE_ES_TaskMain(), CFE_ES_WaitForStartupSync(), CFE_EVS_TaskMain(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.50.3.41 CFE_ES_WriteToSysLog()

```
int32 CFE_ES_WriteToSysLog (
            const char * SpecStringPtr,
             ...  )
```

Description

This routine writes a formatted string to the cFE system log. This can be used to record very low-level errors that can't be reported using the Event Services. This function is used in place of printf for flight software. It should be used for significant startup events, critical errors, and conditionally compiled debug software.

Assumptions, External Events, and Notes:

None

Parameters

in	*SpecStringPtr*	The format string for the log message. This is similar to the format string for a printf() call.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_SYS_LOG_FULL	The cFE system Log is full. This error means the message was not logged at all

Returns

See also

Referenced by CFE_ES_AppCreate(), CFE_ES_CDS_EarlyInit(), CFE_ES_CDS_ValidateAppID(), CFE_ES_CleanUpApp(), CFE_ES_CreateObjects(), CFE_ES_GetCDSBlock(), CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_InitCDSRegistry(), CFE_ES_InitializeCDS(), CFE_ES_InitializeFileSystems(), CFE_ES_LoadLibrary(), CFE_ES_Main(), CFE_ES_ParseFileEntry(), CFE_ES_PerfLogAdd(), CFE_ES_PoolCreateEx(), CFE_ES_PutCDSBlock(), CFE_ES_RebuildCDS(), CFE_ES_RegisterCDSEx(), CFE_ES_ShellOutputCommand(), CFE_ES_StartApplications(), CFE_ES_TaskInit(), CFE_ES_TaskMain(), CFE_ES_UpdateCDSRegistry(), CFE_ES_ValidateCDS(), CFE_EVS_EarlyInit(), CFE_EVS_TaskInit(), CFE_EVS_TaskMain(), CFE_SB_AppInit(), CFE_SB_EarlyInit(), CFE_SB_InitBuffers(), CFE_SB_InitMsgMap(), CFE_SB_LockSharedData(), CFE_SB_TaskMain(), CFE_SB_UnlockSharedData(), and EVS_NotRegistered().

13.51 cfe/fsw/cfe-core/src/inc/cfe_es_events.h File Reference

Macros

- #define CFE_ES_MAX_EID 92

- #define CFE_ES_INIT_INF_EID 1 /* start up message "informational" */

  'cFE ES Initialized'
- #define CFE_ES_INITSTATS_INF_EID 2

  'cFE Version %d.%d.%d chksm %d, OSAL Version %d.%d'
- #define CFE_ES_NOOP_INF_EID 3 /* processed command "informational" */

  'No-op command'
- #define CFE_ES_RESET_INF_EID 4

  'Reset Counters command'
- #define CFE_ES_SHELL_INF_EID 5

  'Invoked shell command %s'
- #define CFE_ES_START_INF_EID 6

  'Started %s from %s, AppID = %d'
- #define CFE_ES_STOP_DBG_EID 7

  'Stop Application %s Initiated.'
- #define CFE_ES_STOP_INF_EID 8

  'Stop Application %s Completed.'
- #define CFE_ES_RESTART_APP_DBG_EID 9

  'Restart Application %s Initiated.'
- #define CFE_ES_RESTART_APP_INF_EID 10

  'Restart Application %s Completed.'
- #define CFE_ES_RELOAD_APP_DBG_EID 11

  'Reload Application %s Initiated.'
- #define CFE_ES_RELOAD_APP_INF_EID 12

  'Reload Application %s Completed.'
- #define CFE_ES_EXIT_APP_INF_EID 13

  'Exit Application %s Completed.'
- #define CFE_ES_ERREXIT_APP_INF_EID 14

  'Exit Application %s Completed.'
- #define CFE_ES_ONE_APP_EID 15

  'Sent %s application data'
- #define CFE_ES_ALL_APPS_EID 16

  'App Info file written to %s, Entries=%d, FileSize=%d'
- #define CFE_ES_SYSLOG1_INF_EID 17

  'Cleared Executive Services log data'
- #define CFE_ES_SYSLOG2_EID 18

  '%s written:Size=%d,Entries=%d'
- #define CFE_ES_ERLOG1_INF_EID 19

  'Cleared mode log data'
- #define CFE_ES_ERLOG2_EID 20

  '%s written:Size=%d'
- #define CFE_ES_MID_ERR_EID 21 /* invalid command packet "error" */

  'Invalid command pipe message ID: 0x%X'
- #define CFE_ES_CC1_ERR_EID 22

  'Invalid ground command code: ID = 0x%X, CC = %d'
- #define CFE_ES_LEN_ERR_EID 23

  'Invalid cmd length: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'
- #define CFE_ES_BOOT_ERR_EID 24 /* command specific "error" */

'Invalid cFE restart type %d'

- #define CFE_ES_SHELL_ERR_EID 25

  'Failed to invoke shell command %s, rc = %08X'
- #define CFE_ES_START_ERR_EID 26

  'Failed to start %s from %s, RC = %08X'
- #define CFE_ES_START_INVALID_FILENAME_ERR_EID 27

  'CFE_ES_StartAppCmd: invalid filename: %s'
- #define CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID 28

  'CFE_ES_StartAppCmd: App Entry Point is NULL.'
- #define CFE_ES_START_NULL_APP_NAME_ERR_EID 29

  'CFE_ES_StartAppCmd: App Name is NULL.'
- #define CFE_ES_START_STACK_ERR_EID 30

  'CFE_ES_StartAppCmd: Stack size is less than system Minimum: %d.'
- #define CFE_ES_START_PRIORITY_ERR_EID 31

  'CFE_ES_StartAppCmd: Priority is too large: %d.'
- #define CFE_ES_START_EXC_ACTION_ERR_EID 32

  'CFE_ES_StartAppCmd: Invalid Exception Action: %d.'
- #define CFE_ES_ERREXIT_APP_ERR_EID 33

  'Exit Application %s on Error Failed: CleanUpApp Error 0x%08X.'
- #define CFE_ES_STOP_ERR1_EID 35

  'Stop Application %s Failed, RC = 0x%08X'
- #define CFE_ES_STOP_ERR2_EID 36

  'Stop Application %s, GetAppIDByName failed. RC = 0x%08X.'
- #define CFE_ES_STOP_ERR3_EID 37

  'Stop Application %s Failed: CleanUpApp Error 0x%08X.'
- #define CFE_ES_RESTART_APP_ERR1_EID 38

  'Restart Application %s Failed, RC = 0x%08X'
- #define CFE_ES_RESTART_APP_ERR2_EID 39

  'Restart Application %s, GetAppIDByName failed. RC = 0x%08X.'
- #define CFE_ES_RESTART_APP_ERR3_EID 40

  'Restart Application %s Failed: AppCreate Error 0x%08X.'
- #define CFE_ES_RESTART_APP_ERR4_EID 41

  'Restart Application %s Failed: CleanUpApp Error 0x%08X.'
- #define CFE_ES_RELOAD_APP_ERR1_EID 42

  'Failed to reload Application %s, rc = %08X'
- #define CFE_ES_RELOAD_APP_ERR2_EID 43

  'Reload Application %s, GetAppIDByName failed. RC = 0x%08X.'
- #define CFE_ES_RELOAD_APP_ERR3_EID 44

  'Reload Application %s Failed: AppCreate Error 0x%08X.'
- #define CFE_ES_RELOAD_APP_ERR4_EID 45

  'Reload Application %s Failed: CleanUpApp Error 0x%08X.'
- #define CFE_ES_EXIT_APP_ERR_EID 46

  'Exit Application %s Failed: CleanUpApp Error 0x%08X.'
- #define CFE_ES_PCR_ERR1_EID 47

  'ES_ProcControlReq: Invalid State (EXCEPTION) Application %s.'
- #define CFE_ES_PCR_ERR2_EID 48

  'ES_ProcControlReq: Unknown State (%d) Application %s.'

- #define CFE_ES_ONE_ERR_EID 49

  'Failed to send %s application data, RC = %08X'
- #define CFE_ES_ONE_APPID_ERR_EID 50

  'Failed to send %s application data: GetAppIDByName Failed, RC = 0x%08X'
- #define CFE_ES_OSCREATE_ERR_EID 51

  'Failed to write App Info file, OS_creat returned %d'
- #define CFE_ES_WRHDR_ERR_EID 52

  'Failed to write App Info file, WriteHdr rtnd %08X, exp %d'
- #define CFE_ES_TASKWR_ERR_EID 53

  'Failed to write App Info file, Task write RC = 0x%08X, exp %d'
- #define CFE_ES_SYSLOG2_ERR_EID 55

  'Error creating file %s, stat=0x%x'
- #define CFE_ES_ERLOG2_ERR_EID 56

  'Error creating file %s, stat=0x%x'
- #define CFE_ES_PERF_STARTCMD_EID 57

  'Start collecting performance data command, trigger mode = d'
- #define CFE_ES_PERF_STARTCMD_ERR_EID 58

  'Cannot start collecting performance data,perf data write in progress'
- #define CFE_ES_PERF_STARTCMD_TRIG_ERR_EID 59

  'Cannot start collecting performance data, trigger mode (d) out of range (d to d)'
- #define CFE_ES_PERF_STOPCMD_EID 60

  'Perf Stop Cmd Rcvd,%s will write %d entries.%dmS dly every %d entries'
- #define CFE_ES_PERF_STOPCMD_ERR1_EID 61

  'Stop performance data cmd,Error creating child task RC=0x%08X'
- #define CFE_ES_PERF_STOPCMD_ERR2_EID 62

  'Stop performance data cmd ignored,perf data write in progress'
- #define CFE_ES_PERF_FILTMSKCMD_EID 63

  'Set Performance Filter Mask command'
- #define CFE_ES_PERF_FILTMSKERR_EID 64

  'Error:Performance Filter Mask Index value greater than CFE_ES_PERF_32BIT_WORDS_IN_MASK (which is a whole number derived from CFE_PLATFORM_ES_PERF_MAX_IDS / 32)'
- #define CFE_ES_PERF_TRIGMSKCMD_EID 65

  'Set Performance Trigger Mask command'
- #define CFE_ES_PERF_TRIGMSKERR_EID 66

  'Error: Performance Trigger Mask Index value greater than CFE_ES_PERF_32BIT_WORDS_IN_MASK (which is a whole number derived from CFE_PLATFORM_ES_PERF_MAX_IDS / 32)'
- #define CFE_ES_PERF_LOG_ERR_EID 67

  'Error creating file %s, stat=%d'
- #define CFE_ES_PERF_DATAWRITTEN_EID 68

  '%s written:Size=%d,EntryCount=%d'
- #define CFE_ES_CDS_REGISTER_ERR_EID 69

  '%s Failed to Register CDS '%s', Status=0x%08X'
- #define CFE_ES_SYSLOGMODE_EID 70

  'Set OverWriteSysLog Command Received with Mode setting = %d'
- #define CFE_ES_ERR_SYSLOGMODE_EID 71

'Set OverWriteSysLog Command: Invalid Mode setting = %d'

- #define CFE_ES_RESET_PR_COUNT_EID 72

  'Reset Processor Reset Count to Zero'
- #define CFE_ES_SET_MAX_PR_COUNT_EID 73

  'Maximum Processor Reset Count set to: %d'
- #define CFE_ES_FILEWRITE_ERR_EID 74

  'File write,byte cnt err,file %s,request=%d,actual=%d'
- #define CFE_ES_RST_ACCESS_EID 75

  'Error accessing ER Log,%s not written.Stat=0x%08x'
- #define CFE_ES_CDS_DELETE_ERR_EID 76

  'Error while deleting '%s' from CDS, See SysLog.(Err=0x%08X)'
- #define CFE_ES_CDS_NAME_ERR_EID 77

  'Unable to locate '%s' in CDS Registry'
- #define CFE_ES_CDS_DELETED_INFO_EID 78

  'Successfully removed '%s' from CDS'
- #define CFE_ES_CDS_DELETE_TBL_ERR_EID 79

  'CDS '%s' is a Critical Table CDS. Must be deleted via TBL Command'
- #define CFE_ES_CDS_OWNER_ACTIVE_EID 80

  'CDS '%s' not deleted because owning app is active'
- #define CFE_ES_TLM_POOL_STATS_INFO_EID 81

  'Successfully telemetered memory pool stats for 0x%08X'
- #define CFE_ES_INVALID_POOL_HANDLE_ERR_EID 82

  'Cannot telemeter memory pool stats. Illegal Handle (0x%08X)'
- #define CFE_ES_CDS_REG_DUMP_INF_EID 83

  'Successfully dumped CDS Registry to '%s':Size=%d,Entries=%d'
- #define CFE_ES_CDS_DUMP_ERR_EID 84

  'Error writing CDS Registry to '%s', Status=0x%08X'
- #define CFE_ES_WRITE_CFE_HDR_ERR_EID 85

  'Error writing cFE File Header to '%s', Status=0x%08X'
- #define CFE_ES_CREATING_CDS_DUMP_ERR_EID 86

  'Error creating CDS dump file '%s', Status=0x%08X'
- #define CFE_ES_TASKINFO_EID 87

  'Task Info file written to %s, Entries=%d, FileSize=%d'
- #define CFE_ES_TASKINFO_OSCREATE_ERR_EID 88

  'Failed to write Task Info file, OS_creat returned %d'
- #define CFE_ES_TASKINFO_WRHDR_ERR_EID 89

  'Failed to write Task Info file, WriteHdr rtnd %08X, exp %d'
- #define CFE_ES_TASKINFO_WR_ERR_EID 90

  'Failed to write Task Info file, Task write RC = 0x%08X, exp %d'
- #define CFE_ES_VERSION_INF_EID 91

  'Mission s.s, s, s'
- #define CFE_ES_BUILD_INF_EID 92

  'Build s s'

13.51.1 Macro Definition Documentation

13.51.1.1 CFE_ES_ALL_APPS_EID

```
#define CFE_ES_ALL_APPS_EID 16
```

Event Message `'App Info file written to %s, Entries=%d, FileSize=%d'`

Type: DEBUG

Cause:

This event message is issued upon successful completion of the cFE Executive Services Query All Applications command

The `'s'` field identifies the name of the file to which all Executive Services Application data has been written. The `Entries` field identifies, in decimal, the number of Applications whose data was written and the `FileSize` field gives the total number of bytes written to the file.

Definition at line 301 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllCmd().

13.51.1.2 CFE_ES_BOOT_ERR_EID

```
#define CFE_ES_BOOT_ERR_EID 24 /* command specific "error" */
```

Event Message `'Invalid cFE restart type %d'`

Type: ERROR

Cause:

This event message is issued when the cFE Executive Services receives a cFE Restart Command whose parameter identifying the restart type is not equal to either CFE_PSP_RST_TYPE_PROCESSOR or CFE_PSP_RST_TYPE_POWERON.

The 'd' field identifies the numeric, in decimal, of the restart type found in the received cFE Restart Command Packet.

Definition at line 433 of file cfe_es_events.h.

Referenced by CFE_ES_RestartCmd().

13.51.1.3 CFE_ES_BUILD_INF_EID

```
#define CFE_ES_BUILD_INF_EID 92
```

Event Message `'Build s s'`

Type: INFORMATION

Cause:

This event message is always automatically issued when the Executive Services Task completes its Initialization, and as part of the Noop command.

The `Build` field identifies the build date, time, hostname and user identifier of the build host machine for the current running binary. The first string is the build date/time, and the second string is formatted as "user@hostname"

By default, if not specified/overridden, the default values of these variables will be: BUILDDATE ==> the output of "date +%Y%m%d%H%M" HOSTNAME ==> the output of "hostname" USER ==> the output of "whoami"

The values can be overridden by setting an environment variable with the names above to the value desired for the field when running "make".

Definition at line 1538 of file cfe_es_events.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.51.1.4 CFE_ES_CC1_ERR_EID

```
#define CFE_ES_CC1_ERR_EID 22
```

Event Message `'Invalid ground command code:  ID = 0x%X, CC = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_ES_CMD_MID message ID has arrived but whose Command Code is not one of the command codes specified in cfe_es.h . This problem is most likely to occur when:

1. A Message ID meant for another Application became corrupted and was set equal to CFE_ES_CMD_MID.
2. The Command Code field in the Message became corrupted.
3. The command database at the ground station has been corrupted.

The `ID` field in the event message specifies the Message ID (in hex) and the `CC` field specifies the Command Code (in decimal) found in the message.

Definition at line 398 of file cfe_es_events.h.

Referenced by CFE_ES_TaskPipe().

13.51.1.5 CFE_ES_CDS_DELETE_ERR_EID

```
#define CFE_ES_CDS_DELETE_ERR_EID 76
```

Event Message `'Error while deleting '%s' from CDS, See SysLog.(Err=0x%08X)'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Delete CDS Command fails to cleanly remove the specified CDS.

The `'s'` field identifies the name of the CDS that was attempted to be deleted the `Err` field specifies, in hex, the error code.

Definition at line 1266 of file cfe_es_events.h.

Referenced by CFE_ES_DeleteCDSCmd().

13.51.1.6 CFE_ES_CDS_DELETE_TBL_ERR_EID

```
#define CFE_ES_CDS_DELETE_TBL_ERR_EID 79
```

Event Message `'CDS '%s' is a Critical Table CDS. Must be deleted via TBL Command'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Delete CDS Command specifies a name for a CDS that is a Critical Table image. Critical Table images can only be deleted via a Table Services command (CFE_TBL_DELE↩TE_CDS_CC).

The `'s'` field identifies the name of the CDS that was attempted to be deleted.

Definition at line 1313 of file cfe_es_events.h.

Referenced by CFE_ES_DeleteCDSCmd().

13.51.1.7 CFE_ES_CDS_DELETED_INFO_EID

```
#define CFE_ES_CDS_DELETED_INFO_EID 78
```

Event Message `'Successfully removed '%s' from CDS'`

Type: INFORMATION

Cause:

This event message is generated when an Executive Services Delete CDS Command is successfully completed.

The `'s'` field identifies the name of the CDS that was deleted.

Definition at line 1296 of file cfe_es_events.h.

Referenced by CFE_ES_DeleteCDSCmd().

13.51.1.8 CFE_ES_CDS_DUMP_ERR_EID

```
#define CFE_ES_CDS_DUMP_ERR_EID 84
```

Event Message `'Error writing CDS Registry to '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Critical Data Store Registry Command was being performed and it encountered a filesystem write error while writing a CDS Registry record.

The `'s'` field identifies the CDS Registry Dump Filename. The `'08X'` field identifies the error code returned from OS_write that caused the command to abort.

Definition at line 1399 of file cfe_es_events.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

13.51.1.9 CFE_ES_CDS_NAME_ERR_EID

```
#define CFE_ES_CDS_NAME_ERR_EID 77
```

Event Message `'Unable to locate '%s' in CDS Registry'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Delete CDS Command specifies a name for a CDS that cannot be found in the CDS Registry.

The `'s'` field identifies the name of the CDS that was attempted to be deleted.

Definition at line 1281 of file cfe_es_events.h.

Referenced by CFE_ES_DeleteCDSCmd().

13.51.1.10 CFE_ES_CDS_OWNER_ACTIVE_EID

```
#define CFE_ES_CDS_OWNER_ACTIVE_EID 80
```

Event Message `'CDS '%s' not deleted because owning app is active'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Delete CDS Command specifies a name for a CDS whose prefix name identifies an application that is still registered in the system. CDSs can only be deleted when their owning applications have been removed from the system.

The `'s'` field identifies the name of the CDS that was attempted to be deleted.

Definition at line 1331 of file cfe_es_events.h.

Referenced by CFE_ES_DeleteCDSCmd().

13.51.1.11 CFE_ES_CDS_REG_DUMP_INF_EID

```
#define CFE_ES_CDS_REG_DUMP_INF_EID 83
```

Event Message `'Successfully dumped CDS Registry to '%s':Size=%d,Entries=%d'`

Type: DEBUG

Cause:

This event message is generated when an Executive Services Dump Critical Data Store Registry Command is successfully executed. The specified file should have been created and contains the CDS Registry Entries.

The `'s'` field identifies the CDS Registry Dump Filename. The first `'d'` field specifies the size of the file (in bytes) The second `'d'` field specifies the number of CDS Registry Records that were written

Definition at line 1382 of file cfe_es_events.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

13.51.1.12 CFE_ES_CDS_REGISTER_ERR_EID

```
#define CFE_ES_CDS_REGISTER_ERR_EID 69
```

Event Message `'%s Failed to Register CDS '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated whenever an Application calls the CFE_ES_RegisterCDS API and fails to successfully create the desired CDS.

The first `'s'` field identifies the name of the Application which made the API call, the second `'s'` field specifies the name of the CDS as requested by the Application and the `Status` field provides the error code which identifies in more detail the nature of the failure (See return codes for the CFE_ES_RegisterCDS API).

Definition at line 1162 of file cfe_es_events.h.

Referenced by CFE_ES_RegisterCDS().

13.51.1.13 CFE_ES_CREATING_CDS_DUMP_ERR_EID

```
#define CFE_ES_CREATING_CDS_DUMP_ERR_EID 86
```

Event Message `'Error creating CDS dump file '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Critical Data Store Registry Command is unable to create the specified file on the onboard filesystem.

The `'s'` field identifies the CDS Registry Dump Filename. The `'08X'` field identifies error code returned by the API OS_creat.

Definition at line 1432 of file cfe_es_events.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

13.51.1.14 CFE_ES_ERLOG1_INF_EID

```
#define CFE_ES_ERLOG1_INF_EID 19
```

Event Message `'Cleared mode log data'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of the cFE Executive Services Clear Exception Reset Log command

Definition at line 342 of file cfe_es_events.h.

Referenced by CFE_ES_ClearERLogCmd().

13.51.1.15 CFE_ES_ERLOG2_EID

```
#define CFE_ES_ERLOG2_EID 20
```

Event Message `'%s written:Size=%d'`

Type: DEBUG

Cause:

This event message is generated when the Exception Reset Log has been successfully written to a file after receiving the cFE Executive Services Write Executive Services Exception Reset Log command

The `'s'` field identifies the name of the file written to and the `Size` field specifies, in decimal, the number of bytes written to the file.

Definition at line 358 of file cfe_es_events.h.

Referenced by CFE_ES_ERLogDump().

13.51.1.16 CFE_ES_ERLOG2_ERR_EID

```
#define CFE_ES_ERLOG2_ERR_EID 56
```

Event Message `'Error creating file %s, stat=0x%x'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Exception Reset Log Command fails while attempting to create the specified file.

The `'s'` field identifies the name of the file that was attempted to be created and the `stat` field specifies, in hex, the error code returned by the OS_creat API.

Definition at line 950 of file cfe_es_events.h.

Referenced by CFE_ES_ERLogDump().

13.51.1.17 CFE_ES_ERR_SYSLOGMODE_EID

```
#define CFE_ES_ERR_SYSLOGMODE_EID 71
```

Event Message `'Set OverWriteSysLog Command:  Invalid Mode setting = %d'`

Type: ERROR

Cause:

This event message is generated upon unsuccessful completion of an Executive Services Set System Log Overwrite Mode Command .

The `setting` field identifies the illegal Overwrite Mode found in the command message. The mode must be either CFE_ES_LogMode_OVERWRITE (0) or CFE_ES_LogMode_DISCARD (1).

Definition at line 1192 of file cfe_es_events.h.

Referenced by CFE_ES_OverWriteSyslogCmd().

13.51.1.18 CFE_ES_ERREXIT_APP_ERR_EID

```
#define CFE_ES_ERREXIT_APP_ERR_EID 33
```

Event Message `'Exit Application %s on Error Failed:  CleanUpApp Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated when ES is completing the processing of the CFE_ES_ExitApp API call with the CFE_ES_RunStatus_APP_ERROR parameter and the call to CFE_ES_CleanUpApp fails. At this point the Application will likely be stopped or deleted, but it may be in an unknown state.

The `'s'` field identifies the name of the Application which was attempted to be reloaded and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 587 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.19 CFE_ES_ERREXIT_APP_INF_EID

```
#define CFE_ES_ERREXIT_APP_INF_EID 14
```

Event Message `'Exit Application %s Completed.'`

Type: INFORMATION

Cause:

This event message is issued when the cFE finishes exiting/cleaning up an application that called the CFE_ES_ExitApp API with an ERROR condition. When an App calls this API, with the CFE_ES_RunStatus_APP_ERROR parameter, it indicates that the Application exited due to an error condition. The details of the error that occurred should be given by the Application through an event message, System Log entry, or both. The request is recorded and the Executive Services App will actually delete cFE Application before issuing this event message.

The `'s'` field identifies the name of the Application that was exited.

Definition at line 268 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.20 CFE_ES_EXIT_APP_ERR_EID

```
#define CFE_ES_EXIT_APP_ERR_EID 46
```

Event Message `'Exit Application %s Failed:  CleanUpApp Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated when ES is completing the processing of the CFE_ES_ExitApp API call and the call to CFE_ES_CleanUpApp fails. At this point the Application will likely be stopped or deleted, but it may be in an unknown state.

The `'s'` field identifies the name of the Application which was attempted to be reloaded and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 810 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.21 CFE_ES_EXIT_APP_INF_EID

```
#define CFE_ES_EXIT_APP_INF_EID 13
```

Event Message `'Exit Application %s Completed.'`

Type: INFORMATION

Cause:

This event message is issued when the cFE finishes exiting/cleaning up an application that called the CFE_ES_ExitApp API with the CFE_ES_RunStatus_APP_EXIT parameter. When an App calls this API, the request is recorded and the Executive Services App will actually delete cFE Application before issuing this event message.

The `'s'` field identifies the name of the Application that was exited.

Definition at line 248 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.22 CFE_ES_FILEWRITE_ERR_EID

```
#define CFE_ES_FILEWRITE_ERR_EID 74
```

Event Message `'File write,byte cnt err,file %s,request=%d,actual=%d'`

Type: ERROR

Cause:

This event message is generated in response to any command requesting information to be written to a file and whose data is not completely written to the specified file.

The `file` field identifies the filename of the file to which the data failed to write completely, the `request` field specifies, in decimal, the number of bytes that were attempted to be written and the `actual` field indicates, in decimal, the actual number of bytes written to the file.

Definition at line 1235 of file cfe_es_events.h.

Referenced by CFE_ES_FileWriteByteCntErr().

13.51.1.23 CFE_ES_INIT_INF_EID

```
#define CFE_ES_INIT_INF_EID 1 /* start up message "informational" */
```

Event Message `'cFE ES Initialized'`

Type: INFORMATION

Cause:

This event message is always automatically issued when the Executive Services Task completes its Initialization.

Definition at line 62 of file cfe_es_events.h.

Referenced by CFE_ES_TaskInit().

13.51.1.24 CFE_ES_INITSTATS_INF_EID

```
#define CFE_ES_INITSTATS_INF_EID 2
```

Event Message `'cFE Version %d.%d.%d chksm %d, OSAL Version %d.%d'`

Type: INFORMATION

Cause:

This event message is always automatically issued when the Executive Services Task completes its Initialization.

The `Version` field identifies the tagged version for the cFE Build, the `chksm` field provides the 16-bit checksum of the cFE Build and the `OSAL Version` field identifies the version of the OS Abstraction Layer on which this particular version of the cFE was built.

Definition at line 78 of file cfe_es_events.h.

Referenced by CFE_ES_TaskInit().

13.51.1.25 CFE_ES_INVALID_POOL_HANDLE_ERR_EID

```
#define CFE_ES_INVALID_POOL_HANDLE_ERR_EID 82
```

Event Message `'Cannot telemeter memory pool stats.  Illegal Handle (0x%08X)'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Telemeter Memory Statistics Command specifies a memory pool handle that is invalid. A handle is determined to be invalid when any of the following are true:

1. The handle does not contain a value that is an integral multiple of 4
2. The handle does not specify a valid area of memory
3. The handle does not point to an area of memory that contains the handle itself
4. The handle does not point to an area of memory whose Size field is an integral multiple of 4
5. The handle does not point to an area of memory whose End field is equal to the Start plus the Size

The `'08X'` field identifies the handle that was found in the command.

Definition at line 1364 of file cfe_es_events.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

13.51.1.26 CFE_ES_LEN_ERR_EID

```
#define CFE_ES_LEN_ERR_EID 23
```

Event Message `'Invalid cmd length:  ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_ES_CMD_MID message ID has arrived but whose packet length does not match the expected length for the specified command code.

The `ID` field in the event message specifies the Message ID (in hex), the `CC` field specifies the Command Code (in decimal), the `Exp` Len field specified the Expected Length (in decimal), and `Len` specifies the message Length (in decimal) found in the message.

Definition at line 416 of file cfe_es_events.h.

Referenced by CFE_ES_VerifyCmdLength().

13.51.1.27 CFE_ES_MAX_EID

```
#define CFE_ES_MAX_EID 92
```

Definition at line 46 of file cfe_es_events.h.

13.51.1.28 CFE_ES_MID_ERR_EID

```
#define CFE_ES_MID_ERR_EID 21 /* invalid command packet "error" */
```

Event Message `'Invalid command pipe message ID: 0x%X'`

Type: ERROR

Cause:

This event message is generated when a message has arrived on the cFE Executive Services Application's Message Pipe that has a Message ID that is neither CFE_ES_SEND_HK_MID or CFE_ES_CMD_MID. Most likely, the cFE Software Bus routing table has become corrupt and is sending messages targeted for other Applications to the cFE Executive Services Application.

The `ID` field in the event message identifies the message ID (in hex) that was found in the message.

Definition at line 377 of file cfe_es_events.h.

Referenced by CFE_ES_TaskPipe().

13.51.1.29 CFE_ES_NOOP_INF_EID

```
#define CFE_ES_NOOP_INF_EID 3 /* processed command "informational" */
```

Event Message `'No-op command'`

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Executive Services NO-OP command

Definition at line 90 of file cfe_es_events.h.

Referenced by CFE_ES_NoopCmd().

13.51.1.30 CFE_ES_ONE_APP_EID

```
#define CFE_ES_ONE_APP_EID 15
```

Event Message `'Sent %s application data'`

Type: DEBUG

Cause:

This event message is issued upon successful completion of the cFE Executive Services Query One Application command

The `'s'` field identifies the name of the Application whose Executive Services Application information has been telemetered.

Definition at line 284 of file cfe_es_events.h.

Referenced by CFE_ES_QueryOneCmd().

13.51.1.31 CFE_ES_ONE_APPID_ERR_EID

```
#define CFE_ES_ONE_APPID_ERR_EID 50
```

Event Message `'Failed to send %s application data: GetAppIDByName Failed, RC = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Request Application Data Command failed.

The `'s'` field identifies the name of the Application whose data was attempted to be telemetered and the `rc` field identifies the error code, in hex, that may identify the precise reason for the failure.

Definition at line 873 of file cfe_es_events.h.

Referenced by CFE_ES_QueryOneCmd().

13.51.1.32 CFE_ES_ONE_ERR_EID

```
#define CFE_ES_ONE_ERR_EID 49
```

Event Message `'Failed to send %s application data, RC = %08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Request Application Data Command failed.

The `'s'` field identifies the name of the Application whose data was attempted to be telemetered and the `rc` field identifies the error code, in hex, that may identify the precise reason for the failure.

Definition at line 857 of file cfe_es_events.h.

Referenced by CFE_ES_QueryOneCmd().

13.51.1.33 CFE_ES_OSCREATE_ERR_EID

```
#define CFE_ES_OSCREATE_ERR_EID 51
```

Event Message `'Failed to write App Info file, OS_creat returned %d'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Application Data Command fails to create the dump file.

The `'d'` parameter identifies, in decimal, the error code returned by OS_creat when the attempt was made to create the file.

Definition at line 889 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllCmd().

13.51.1.34 CFE_ES_PCR_ERR1_EID

```
#define CFE_ES_PCR_ERR1_EID 47
```

Event Message `'ES_ProcControlReq:  Invalid State (EXCEPTION) Application %s.'`

Type: ERROR

Cause:

This event message is generated when ES is processing it's internal Application table and encounters an App with the EXCEPTION state. Because exceptions are supposed to be processed immediately, this is an invalid state and should not happen. It may indicate some sort of memory corruption or other problem.

Definition at line 824 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.35 CFE_ES_PCR_ERR2_EID

```
#define CFE_ES_PCR_ERR2_EID 48
```

Event Message `'ES_ProcControlReq:  Unknown State ( %d ) Application %s.'`

Type: ERROR

Cause:

This event message is generated when ES is processing it's internal Application table and encounters an App with an unknown state. If this message occurs, it might be an indication of a memory corruption or other problem.

Definition at line 841 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.36 CFE_ES_PERF_DATAWRITTEN_EID

```
#define CFE_ES_PERF_DATAWRITTEN_EID 68
```

Event Message `'%s written:Size=%d,EntryCount=%d'`

Type: DEBUG

Cause:

This event message is generated when the Performance Log has been successfully written to a file after receiving the cFE Executive Services Stop Performance Analyzer Data Collection Command

The `'s'` field identifies the name of the file written to, the `Size` field specifies, in decimal, the number of bytes written to the file and the `EntryCount` field identifies the number of data entries that were written.

Definition at line 1145 of file cfe_es_events.h.

Referenced by CFE_ES_PerfLogDump().

13.51.1.37 CFE_ES_PERF_FILTMSKCMD_EID

```
#define CFE_ES_PERF_FILTMSKCMD_EID 63
```

Event Message `'Set Performance Filter Mask command'`

Type: DEBUG

Cause:

This event message is generated in response to receiving an Executive Services Set Performance Analyzer Filter Mask Command .

Definition at line 1062 of file cfe_es_events.h.

Referenced by CFE_ES_SetPerfFilterMaskCmd().

13.51.1.38 CFE_ES_PERF_FILTMSKERR_EID

```
#define CFE_ES_PERF_FILTMSKERR_EID 64
```

Event Message `'Error:Performance Filter Mask Index value greater than CFE_ES_PE↩ RF_32BIT_WORDS_IN_MASK (which is a whole number derived from CFE_↩ PLATFORM_ES_PERF_MAX_IDS / 32)'`

Type: ERROR

Cause:

This event message is generated in response to receiving an Executive Services Set Performance Analyzer Filter Mask Command .

Definition at line 1079 of file cfe_es_events.h.

Referenced by CFE_ES_SetPerfFilterMaskCmd().

13.51.1.39 CFE_ES_PERF_LOG_ERR_EID

```
#define CFE_ES_PERF_LOG_ERR_EID 67
```

Event Message `'Error creating file %s, stat=%d'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Stop Performance Analyzer Data Collection Command fails to create the associated logic analyzer dump file.

The `'s'` field identifies the name of the file that was attempted to be created and the `stat` field specifies, in decimal, the error code returned by the OS_creat API.

Definition at line 1127 of file cfe_es_events.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.51.1.40 CFE_ES_PERF_STARTCMD_EID

```
#define CFE_ES_PERF_STARTCMD_EID 57
```

Event Message `'Start collecting performance data command, trigger mode = d'`

Type: DEBUG

Cause:

This event message is generated in response to receiving an Executive Services Start Performance Analyzer Data Collection Command

The `'d'` field identifies the requested trigger mode. Valid values are CFE_ES_PERF_TRIGGER_START (0) , CFE_↩ ES_PERF_TRIGGER_CENTER (1) , and CFE_ES_PERF_TRIGGER_END (2)

Definition at line 965 of file cfe_es_events.h.

Referenced by CFE_ES_StartPerfDataCmd().

13.51.1.41 CFE_ES_PERF_STARTCMD_ERR_EID

```
#define CFE_ES_PERF_STARTCMD_ERR_EID 58
```

Event Message `'Cannot start collecting performance data,perf data write in progress'`

Type: ERROR

Cause:

This event message is generated in response to receiving an Executive Services Start Performance Analyzer Data Collection Command

Definition at line 977 of file cfe_es_events.h.

Referenced by CFE_ES_StartPerfDataCmd().

13.51.1.42 CFE_ES_PERF_STARTCMD_TRIG_ERR_EID

```
#define CFE_ES_PERF_STARTCMD_TRIG_ERR_EID 59
```

Event Message `'Cannot start collecting performance data, trigger mode (d) out of range (d to d)'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Start Performance Analyzer Data Collection Command command is received with a bad value for the requested trigger mode.

The first `'d'` field identifies the received trigger mode value. The second and third `'d'` fields specify the valid range of values for the trigger mode. Valid values are CFE_ES_PerfMode_t CFE_ES_PERF_TRIGGER_START (0), CFE_ES_PerfMode_t CFE_ES_PERF_TRIGGER_CENTER (1), and CFE_ES_PerfMode_t CFE_ES_PERF_TRIGGER_END (2).

Definition at line 996 of file cfe_es_events.h.

Referenced by CFE_ES_StartPerfDataCmd().

13.51.1.43 CFE_ES_PERF_STOPCMD_EID

```
#define CFE_ES_PERF_STOPCMD_EID 60
```

Event Message `'Perf Stop Cmd Rcvd,%s will write %d entries.%dmS dly every %d entries'`

Type: DEBUG

Cause:

This event message is generated upon receipt of a successful Performance Data Stop Command after receiving the cFE Executive Services Stop Performance Analyzer Data Collection Command

The `'s'` field identifies the name of the file write task that has begun execution. The first `'d'` identifies the total number of performance entries(in decimal) that will be written to the file. A performance data entry is defined by an unsigned 32 bit data point and an unsigned 64 bit time stamp. The second `'d'` identifies the millisecond delay between writes and the third `'d'` identifies the number of entries written (in decimal) between delays.

Definition at line 1016 of file cfe_es_events.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.51.1.44 CFE_ES_PERF_STOPCMD_ERR1_EID

```
#define CFE_ES_PERF_STOPCMD_ERR1_EID 61
```

Event Message `'Stop performance data cmd,Error creating child task RC=0x%08X'`

Type: ERROR

Cause:

This event message is generated upon receipt of an unsuccessful Performance Data Stop Command after receiving the cFE Executive Services Stop Performance Analyzer Data Collection Command

The `'RC'` field specifies, in hex, the error code returned by the CFE_ES_CreateChildTask API

Definition at line 1033 of file cfe_es_events.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.51.1.45 CFE_ES_PERF_STOPCMD_ERR2_EID

```
#define CFE_ES_PERF_STOPCMD_ERR2_EID 62
```

Event Message `'Stop performance data cmd ignored,perf data write in progress'`

Type: ERROR

Cause:

This event message is generated upon receipt of an unsuccessful Performance Data Stop Command after receiving the cFE Executive Services Stop Performance Analyzer Data Collection Command

Definition at line 1048 of file cfe_es_events.h.

Referenced by CFE_ES_StopPerfDataCmd().

13.51.1.46 CFE_ES_PERF_TRIGMSKCMD_EID

```
#define CFE_ES_PERF_TRIGMSKCMD_EID 65
```

Event Message `'Set Performance Trigger Mask command'`

Type: DEBUG

Cause:

This event message is generated in response to receiving an Executive Services Set Performance Analyzer Trigger Mask Command .

Definition at line 1093 of file cfe_es_events.h.

Referenced by CFE_ES_SetPerfTriggerMaskCmd().

13.51.1.47 CFE_ES_PERF_TRIGMSKERR_EID

```
#define CFE_ES_PERF_TRIGMSKERR_EID 66
```

Event Message `'Error: Performance Trigger Mask Index value greater than CFE_ES_PERF_32BIT_WORDS_IN_MASK (which is a whole number derived from CFE_PLATFORM_ES_PERF_MAX_IDS / 32)'`

Type: ERROR

Cause:

This event message is generated in response to receiving an Executive Services Set Performance Analyzer Trigger Mask Command .

Definition at line 1110 of file cfe_es_events.h.

Referenced by CFE_ES_SetPerfTriggerMaskCmd().

13.51.1.48 CFE_ES_RELOAD_APP_DBG_EID

```
#define CFE_ES_RELOAD_APP_DBG_EID 11
```

Event Message `'Reload Application %s Initiated.'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the cFE Executive Services Reload Application command Note that when this event is displayed, the Application is not reloaded. ES has accepted the request to reload the application, and it will be reloaded after the app exits it's main loop, or times out.

The `'s'` field identifies the name of the Application that will be reloaded.

Definition at line 216 of file cfe_es_events.h.

Referenced by CFE_ES_ReloadAppCmd().

13.51.1.49 CFE_ES_RELOAD_APP_ERR1_EID

```
#define CFE_ES_RELOAD_APP_ERR1_EID 42
```

Event Message `'Failed to reload Application %s, rc = %08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Reload Application Command fails.

The `'s'` field identifies the name of the Application which was attempted to be reloaded and the `rc` field identifies the error code, in hex, that may identify the precise reason for the failure.

Definition at line 736 of file cfe_es_events.h.

Referenced by CFE_ES_ReloadAppCmd().

13.51.1.50 CFE_ES_RELOAD_APP_ERR2_EID

```
#define CFE_ES_RELOAD_APP_ERR2_EID 43
```

Event Message `'Reload Application %s, GetAppIDByName failed.  RC = 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Reload Application Command which fails. This message is for a specific failure when the call to CFE_ES_GetAppIDByName fails. The application will not be reloaded at this point.

The `'s'` field identifies the name of the Application which was attempted to be reloaded and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 754 of file cfe_es_events.h.

Referenced by CFE_ES_ReloadAppCmd().

13.51.1.51 CFE_ES_RELOAD_APP_ERR3_EID

```
#define CFE_ES_RELOAD_APP_ERR3_EID 44
```

Event Message `'Reload Application %s Failed:  AppCreate Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Reload Application Command which fails. This message is for a specific failure when the call to CFE_ES_AppCreate fails. The application will not be reloaded at this point.

The `'s'` field identifies the name of the Application which was attempted to be reloaded and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 773 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.52 CFE_ES_RELOAD_APP_ERR4_EID

```
#define CFE_ES_RELOAD_APP_ERR4_EID 45
```

Event Message `'Reload Application %s Failed:  CleanUpApp Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Reload Application Command which fails. This message is for a specific failure when the call to CFE_ES_CleanUpApp fails. The application will not be reloaded at this point, and will likely be deleted or in an unknown state.

The `'s'` field identifies the name of the Application which was attempted to be reloaded and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 792 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.53 CFE_ES_RELOAD_APP_INF_EID

```
#define CFE_ES_RELOAD_APP_INF_EID 12
```

Event Message `'Reload Application %s Completed.'`

Type: INFORMATION

Cause:

This event message is issued when the cFE finishes Reloading the cFE Application That was started when the Restart Application command was issued.

The `'s'` field identifies the name of the Application that was reloaded.

Definition at line 232 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.54 CFE_ES_RESET_INF_EID

```
#define CFE_ES_RESET_INF_EID 4
```

Event Message `'Reset Counters command'`

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Executive Services Reset Counters command

Definition at line 102 of file cfe_es_events.h.

Referenced by CFE_ES_ResetCountersCmd().

13.51.1.55 CFE_ES_RESET_PR_COUNT_EID

```
#define CFE_ES_RESET_PR_COUNT_EID 72
```

Event Message `'Reset Processor Reset Count to Zero'`

Type: INFORMATION

Cause:

This event message is always generated in response to the Executive Services Set Processor Reset Counter to Zero Command .

Definition at line 1204 of file cfe_es_events.h.

Referenced by CFE_ES_ResetPRCountCmd().

13.51.1.56 CFE_ES_RESTART_APP_DBG_EID

```
#define CFE_ES_RESTART_APP_DBG_EID 9
```

Event Message `'Restart Application %s Initiated.'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the cFE Executive Services Restart Application command Note that when this event is displayed, the Application is not restarted. ES has accepted the request to restart the application, and it will be restarted after the app exits it's main loop, or times out.

The `'s'` field identifies the name of the Application that will be restarted.

Definition at line 182 of file cfe_es_events.h.

Referenced by CFE_ES_RestartAppCmd().

13.51.1.57 CFE_ES_RESTART_APP_ERR1_EID

```
#define CFE_ES_RESTART_APP_ERR1_EID 38
```

Event Message `'Restart Application %s Failed, RC = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Restart Application Command fails.

The `'s'` field identifies the name of the Application which was attempted to be reset and the `rc` field identifies the error code, in hex, that may identify the precise reason for the failure.

Definition at line 660 of file cfe_es_events.h.

Referenced by CFE_ES_RestartAppCmd().

13.51.1.58 CFE_ES_RESTART_APP_ERR2_EID

```
#define CFE_ES_RESTART_APP_ERR2_EID 39
```

Event Message `'Restart Application %s, GetAppIDByName failed.  RC = 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Restart Application Command which fails. This message is for a specific failure when the call to CFE_ES_GetAppIDByName fails. The application will not be restarted at this point.

The `'s'` field identifies the name of the Application which was attempted to be restarted and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 678 of file cfe_es_events.h.

Referenced by CFE_ES_RestartAppCmd().

13.51.1.59 CFE_ES_RESTART_APP_ERR3_EID

```
#define CFE_ES_RESTART_APP_ERR3_EID 40
```

Event Message `'Restart Application %s Failed:  AppCreate Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Restart Application Command which fails. This message is for a specific failure when the call to CFE_ES_AppCreate fails. The application will not be restarted at this point.

The `'s'` field identifies the name of the Application which was attempted to be restarted and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 700 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.60 CFE_ES_RESTART_APP_ERR4_EID

```
#define CFE_ES_RESTART_APP_ERR4_EID 41
```

Event Message `'Restart Application %s Failed:  CleanUpApp Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Restart Application Command which fails. This message is for a specific failure when the call to CFE_ES_CleanUpApp fails. The application will not be restarted at this point, but will likely be deleted or in an unknown state.

The `'s'` field identifies the name of the Application which was attempted to be restarted and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 719 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.61 CFE_ES_RESTART_APP_INF_EID

```
#define CFE_ES_RESTART_APP_INF_EID 10
```

Event Message `'Restart Application %s Completed.'`

Type: INFORMATION

Cause:

This event message is issued when the cFE finishes Restarting the cFE Application That was started when the Restart Application command was issued.

The `'s'` field identifies the name of the Application that was reloaded.

Definition at line 197 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.62 CFE_ES_RST_ACCESS_EID

```
#define CFE_ES_RST_ACCESS_EID 75
```

Event Message `'Error accessing ER Log,%s not written.Stat=0x%08x'`

Type: ERROR

Cause:

This event message is generated in response to an Exception Reset Log Dump command and there is an error obtaining the contents of the ER Log.

The `'s'` field identifies the filename of the file to which the data failed to write, the `Stat` field specifies, in hex, the error status returned from CFE_PSP_GetResetArea.

Definition at line 1250 of file cfe_es_events.h.

Referenced by CFE_ES_ERLogDump().

13.51.1.63 CFE_ES_SET_MAX_PR_COUNT_EID

```
#define CFE_ES_SET_MAX_PR_COUNT_EID 73
```

Event Message `'Maximum Processor Reset Count set to:  %d'`

Type: INFORMATION

Cause:

This event message is always generated in response to the Executive Services Set Maximum Processor Reset Limit Command .

The `'d'` field identifies, in decimal, the number of Processor Resets that will need to occur before a Power-On Reset is automatically performed.

Definition at line 1219 of file cfe_es_events.h.

Referenced by CFE_ES_SetMaxPRCountCmd().

13.51.1.64 CFE_ES_SHELL_ERR_EID

```
#define CFE_ES_SHELL_ERR_EID 25
```

Event Message `'Failed to invoke shell command %s, rc = %08X'`

Type: ERROR

Cause:

This event message is generated whenever the cFE Executive Services receives an OS Shell command, via the Executive Services Shell Command , and the underlying OS returns an error code.

The 's' field in the message identifies the shell command string that was issued and the `rc` field displays the shell's return code, in hex.

Definition at line 449 of file cfe_es_events.h.

Referenced by CFE_ES_ShellCmd().

13.51.1.65 CFE_ES_SHELL_INF_EID

```
#define CFE_ES_SHELL_INF_EID 5
```

Event Message `'Invoked shell command %s'`

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Executive Services Shell Command

The `'s'` string contains the actual shell command string issued.

Definition at line 116 of file cfe_es_events.h.

Referenced by CFE_ES_ShellCmd().

13.51.1.66 CFE_ES_START_ERR_EID

```
#define CFE_ES_START_ERR_EID 26
```

Event Message `'Failed to start %s from %s, RC = %08X'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message is a general failure when the command passes the parameter validation, but fails when a call to CFE_↩ ES_AppCreate is called.

The `'s'` term identifies the name of the Application that was attempted to start. The second `'s'` field specifies the file from which the Application was loaded. The `'X'` field is the return code returned by the CFE_ES_AppCreate.

Definition at line 468 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.67 CFE_ES_START_EXC_ACTION_ERR_EID

```
#define CFE_ES_START_EXC_ACTION_ERR_EID 32
```

Event Message `'CFE_ES_StartAppCmd: Invalid Exception Action: %d.'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message reports a command failure when the Application Exception Action parameter is invalid. The valid options for this parameter are: 0 = Application will restart on an exception 1 = Application cause a processor restart on exception.

The `'d'` term identifies the Exception Action parameter that was given in the command.

Definition at line 570 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.68 CFE_ES_START_INF_EID

```
#define CFE_ES_START_INF_EID 6
```

Event Message `'Started %s from %s, AppID = %d'`

Type: INFORMATION

Cause:

This event message is automatically issued upon successful completion of a cFE Executive Services Start Application command

The first `'s'` string identifies the name of the started Application, the second `'s'` string identifies the filename from which the Application was loaded and the `AppId` field specifies the Application ID assigned to the newly started Application by the cFE Executive Services.

Definition at line 133 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.69 CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID

```
#define CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID 28
```

Event Message `'CFE_ES_StartAppCmd:  App Entry Point is NULL.'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message reports a command failure when the Start Appplication Command is given a NULL Application Entry Point parameter. The command must contain an application entry point string. (Example: "SC_AppMain").

Definition at line 502 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.70 CFE_ES_START_INVALID_FILENAME_ERR_EID

```
#define CFE_ES_START_INVALID_FILENAME_ERR_EID 27
```

Event Message `'CFE_ES_StartAppCmd: invalid filename: %s'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message reports a command failure when the Start Appplication Command is given an invalid filename. (Either NULL or too short to be a valid cFE file name).

The `'s'` term identifies the invalid filename that was sent with the command.

Definition at line 485 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.71 CFE_ES_START_NULL_APP_NAME_ERR_EID

```
#define CFE_ES_START_NULL_APP_NAME_ERR_EID 29
```

Event Message `'CFE_ES_StartAppCmd: App Name is NULL.'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message reports a command failure when the Start Appplication Command is given a NULL Application Name parameter. The command must contain an application name string.

Definition at line 517 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.72 CFE_ES_START_PRIORITY_ERR_EID

```
#define CFE_ES_START_PRIORITY_ERR_EID 31
```

Event Message `'CFE_ES_StartAppCmd:  Priority is too large:  %d.'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message reports a command failure when the Application priority greater than the maximum priority for a Task defined by the OS Abstraction Layer (256).

The `'d'` term identifies the priority that was given in the command.

Definition at line 551 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.73 CFE_ES_START_STACK_ERR_EID

```
#define CFE_ES_START_STACK_ERR_EID 30
```

Event Message `'CFE_ES_StartAppCmd:  Stack size is less than system Minimum:  %d.'`

Type: ERROR

Cause:

This event message is generated for an error encountered in response to an Executive Services Start Application Command .

This message reports a command failure when the Application Stack Size parameter is less than the default stack size defined in the cfe_platform_cfg.h file: CFE_PLATFORM_ES_DEFAULT_STACK_SIZE.

The `'d'` term identifies the size of the stack that was given in the command.

Definition at line 534 of file cfe_es_events.h.

Referenced by CFE_ES_StartAppCmd().

13.51.1.74 CFE_ES_STOP_DBG_EID

```
#define CFE_ES_STOP_DBG_EID 7
```

Event Message `'Stop Application %s Initiated.'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the cFE Executive Services Stop Application command Note that when this event is displayed, the Application is not deleted. ES has accepted the request to delete the application, and it will be deleted after the app exits it's main loop, or times out.

The `'s'` field identifies the name of the Application that will be stopped.

Definition at line 150 of file cfe_es_events.h.

Referenced by CFE_ES_StopAppCmd().

13.51.1.75 CFE_ES_STOP_ERR1_EID

```
#define CFE_ES_STOP_ERR1_EID 35
```

Event Message `'Stop Application %s Failed, RC = 0x%08X'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Stop Application Command which fails.

The `'s'` field identifies the name of the Application which was attempted to be stopped and the `rc` field identifies the error code, in hex, that may identify the precise reason for the failure.

Definition at line 603 of file cfe_es_events.h.

Referenced by CFE_ES_StopAppCmd().

13.51.1.76 CFE_ES_STOP_ERR2_EID

```
#define CFE_ES_STOP_ERR2_EID 36
```

Event Message `'Stop Application %s, GetAppIDByName failed.  RC = 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Stop Application Command which fails. This message is for a specific failure when the call to CFE_ES_GetAppIDByName fails. The application will not be deleted at this point.

The `'s'` field identifies the name of the Application which was attempted to be stopped and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 621 of file cfe_es_events.h.

Referenced by CFE_ES_StopAppCmd().

13.51.1.77 CFE_ES_STOP_ERR3_EID

```
#define CFE_ES_STOP_ERR3_EID 37
```

Event Message `'Stop Application %s Failed:  CleanUpApp Error 0x%08X.'`

Type: ERROR

Cause:

This event message is generated upon receipt of an Executive Services Stop Application Command which fails. This message is for a specific failure when the call to CFE_ES_GetAppIDByName fails. The application will not be deleted at this point.

The `'s'` field identifies the name of the Application which was attempted to be stopped and the `RC` field identifies the error code, in hex, that will identify the precise reason for the failure.

Definition at line 643 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.78 CFE_ES_STOP_INF_EID

```
#define CFE_ES_STOP_INF_EID 8
```

Event Message `'Stop Application %s Completed.'`

Type: INFORMATION

Cause:

This event message is issued when the cFE finishes deleting the cFE Application That was started when the Stop Application command was issued.

The `'s'` field identifies the name of the Application that was stopped.

Definition at line 165 of file cfe_es_events.h.

Referenced by CFE_ES_ProcessControlRequest().

13.51.1.79 CFE_ES_SYSLOG1_INF_EID

```
#define CFE_ES_SYSLOG1_INF_EID 17
```

Event Message `'Cleared Executive Services log data'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of the cFE Executive Services Clear System Log command

Definition at line 313 of file cfe_es_events.h.

Referenced by CFE_ES_ClearSyslogCmd().

13.51.1.80 CFE_ES_SYSLOG2_EID

```
#define CFE_ES_SYSLOG2_EID 18
```

Event Message `'%s written:Size=%d,Entries=%d'`

Type: DEBUG

Cause:

This event message is generated when the System Log has been successfully written to a file after receiving the cFE Executive Services Write Executive Services System Log command

The `'s'` field identifies the name of the file written to, the `Size` field specifies, in decimal, the number of bytes written to the file and the `Entries` field identifies the number of System Log messages that were written.

Definition at line 330 of file cfe_es_events.h.

Referenced by CFE_ES_SysLogDump().

13.51.1.81 CFE_ES_SYSLOG2_ERR_EID

```
#define CFE_ES_SYSLOG2_ERR_EID 55
```

Event Message `'Error creating file %s, stat=0x%x'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump System Log Command fails while attempting to create the specified file.

The `'s'` field identifies the name of the file that was attempted to be created and the `stat` field specifies, in hex, the error code returned by the OS_creat API.

Definition at line 935 of file cfe_es_events.h.

Referenced by CFE_ES_SysLogDump().

13.51.1.82 CFE_ES_SYSLOGMODE_EID

```
#define CFE_ES_SYSLOGMODE_EID 70
```

Event Message `'Set OverWriteSysLog Command Received with Mode setting = %d'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of an Executive Services Set System Log Overwrite Mode Command .

The `setting` field identifies the newly chosen Overwrite Mode and should be equal to either CFE_ES_LogMode_↩ OVERWRITE or CFE_ES_LogMode_DISCARD.

Definition at line 1177 of file cfe_es_events.h.

Referenced by CFE_ES_OverWriteSyslogCmd().

13.51.1.83 CFE_ES_TASKINFO_EID

```
#define CFE_ES_TASKINFO_EID 87
```

Event Message `'Task Info file written to %s, Entries=%d, FileSize=%d'`

Type: DEBUG

Cause:

This event message is issued upon successful completion of the cFE Executive Services Query All Tasks command

The `'s'` field identifies the name of the file to which all Executive Services Task data has been written. The `Entries` field identifies, in decimal, the number of Tasks whose data was written and the `FileSize` field gives the total number of bytes written to the file.

Definition at line 1450 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllTasksCmd().

13.51.1.84 CFE_ES_TASKINFO_OSCREATE_ERR_EID

```
#define CFE_ES_TASKINFO_OSCREATE_ERR_EID 88
```

Event Message `'Failed to write Task Info file, OS_creat returned %d'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Task Data Command fails to create the dump file.

The `'d'` parameter identifies, in decimal, the error code returned by OS_creat when the attempt was made to create the file.

Definition at line 1466 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllTasksCmd().

13.51.1.85 CFE_ES_TASKINFO_WR_ERR_EID

```
#define CFE_ES_TASKINFO_WR_ERR_EID 90
```

Event Message `'Failed to write Task Info file, Task write RC = 0x%08X, exp %d'`

Type: ERROR

Cause:

This event message is generated whenever an Executive Services Dump Tasks Data Command fails while writing Tasks data to the specified file.

The `rtnd` field contains, in hex, the error code returned from the OS_write API. The expected return value is identified, in decimal, in the `exp` field.

Definition at line 1497 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllTasksCmd().

13.51.1.86 CFE_ES_TASKINFO_WRHDR_ERR_EID

```
#define CFE_ES_TASKINFO_WRHDR_ERR_EID 89
```

Event Message `'Failed to write Task Info file, WriteHdr rtnd %08X, exp %d'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Task Data Command fails while writing the cFE Standard File Header.

The `rtnd` field contains the error code returned by the CFE_FS_WriteHeader API. Nominally, the returned result should have been equal to the `exp` field (i.e. - sizeof(CFE_FS_Header_t)).

Definition at line 1481 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllTasksCmd().

13.51.1.87 CFE_ES_TASKWR_ERR_EID

```
#define CFE_ES_TASKWR_ERR_EID 53
```

Event Message `'Failed to write App Info file, Task write RC = 0x%08X, exp %d'`

Type: ERROR

Cause:

This event message is generated whenever an Executive Services Dump Application Data Command fails while writing Application data to the specified file.

The `rtnd` field contains, in hex, the error code returned from the OS_write API. The expected return value is identified, in decimal, in the `exp` field.

Definition at line 920 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllCmd().

13.51.1.88 CFE_ES_TLM_POOL_STATS_INFO_EID

```
#define CFE_ES_TLM_POOL_STATS_INFO_EID 81
```

Event Message `'Successfully telemetered memory pool stats for 0x%08X'`

Type: DEBUG

Cause:

This event message is generated following successful execution of the Telemeter Memory Statistics Command .

Definition at line 1343 of file cfe_es_events.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

13.51.1.89 CFE_ES_VERSION_INF_EID

```
#define CFE_ES_VERSION_INF_EID 91
```

Event Message `'Mission s.s, s, s'`

Type: INFORMATION

Cause:

This event message is always automatically issued when the Executive Services Task completes its Initialization

The `Mission` field identifies the tagged build identifiers and configuration name. If available, this will also indicate the revision control identifiers for CFE and OSAL that this binary was built with.

Definition at line 1514 of file cfe_es_events.h.

Referenced by CFE_ES_TaskInit().

13.51.1.90 CFE_ES_WRHDR_ERR_EID

```
#define CFE_ES_WRHDR_ERR_EID 52
```

Event Message `'Failed to write App Info file, WriteHdr rtnd %08X, exp %d'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Application Data Command fails while writing the cFE Standard File Header.

The `rtnd` field contains the error code returned by the CFE_FS_WriteHeader API. Nominally, the returned result should have been equal to the `exp` field (i.e. - sizeof(CFE_FS_Header_t)).

Definition at line 904 of file cfe_es_events.h.

Referenced by CFE_ES_QueryAllCmd().

13.51.1.91 CFE_ES_WRITE_CFE_HDR_ERR_EID

```
#define CFE_ES_WRITE_CFE_HDR_ERR_EID 85
```

Event Message `'Error writing cFE File Header to '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Executive Services Dump Critical Data Store Registry Command command successfully created the CDS Dump File onboard but encountered an error while writing the standard cFE File Header to the file.

The `'s'` field identifies the CDS Registry Dump Filename. The `'08X'` field identifies error code returned by the API CFE_FS_WriteHeader.

Definition at line 1416 of file cfe_es_events.h.

Referenced by CFE_ES_DumpCDSRegistryCmd().

13.52 cfe/fsw/cfe-core/src/inc/cfe_es_extern_typedefs.h File Reference

```
#include "common_types.h"
```

Typedefs

- typedef uint8 CFE_ES_LogMode_Enum_t
  Identifies handling of log messages after storage is filled.
- typedef uint8 CFE_ES_ExceptionAction_Enum_t
  Identifies action to take if exception occurs.
- typedef uint8 CFE_ES_AppType_Enum_t
  Identifies type of CFE application.
- typedef uint32 CFE_ES_RunStatus_Enum_t
  Run Status and Exit Status identifiers.
- typedef uint32 CFE_ES_SystemState_Enum_t
  The overall cFE System State.
- typedef uint8 CFE_ES_LogEntryType_Enum_t
  Type of entry in the Error and Reset (ER) Log.
- typedef uint32 CFE_ES_AppState_Enum_t
  Application Run State.

Enumerations

- enum CFE_ES_LogMode { CFE_ES_LogMode_OVERWRITE = 0, CFE_ES_LogMode_DISCARD = 1 }
  Label definitions associated with CFE_ES_LogMode_Enum_t.
- enum CFE_ES_ExceptionAction { CFE_ES_ExceptionAction_RESTART_APP = 0, CFE_ES_ExceptionAction_PROC_RESTART = 1 }
  Label definitions associated with CFE_ES_ExceptionAction_Enum_t.
- enum CFE_ES_AppType { CFE_ES_AppType_CORE = 1, CFE_ES_AppType_EXTERNAL = 2 }
  Label definitions associated with CFE_ES_AppType_Enum_t.
- enum CFE_ES_RunStatus {
  CFE_ES_RunStatus_APP_RUN = 1, CFE_ES_RunStatus_APP_EXIT = 2, CFE_ES_RunStatus_APP_ERROR = 3, CFE_ES_RunStatus_SYS_EXCEPTION = 4,
  CFE_ES_RunStatus_SYS_RESTART = 5, CFE_ES_RunStatus_SYS_RELOAD = 6, CFE_ES_RunStatus_SYS_DELETE = 7, CFE_ES_RunStatus_CORE_APP_INIT_ERROR = 8,
  CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR = 9 }
  Label definitions associated with CFE_ES_RunStatus_Enum_t.
- enum CFE_ES_SystemState {
  CFE_ES_SystemState_UNDEFINED = 0, CFE_ES_SystemState_EARLY_INIT = 1, CFE_ES_SystemState_CORE_STARTUP = 2, CFE_ES_SystemState_CORE_READY = 3,
  CFE_ES_SystemState_APPS_INIT = 4, CFE_ES_SystemState_OPERATIONAL = 5, CFE_ES_SystemState_SHUTDOWN = 6 }
  Label definitions associated with CFE_ES_SystemState_Enum_t.
- enum CFE_ES_LogEntryType { CFE_ES_LogEntryType_CORE = 1, CFE_ES_LogEntryType_APPLICATION = 2 }
  Label definitions associated with CFE_ES_LogEntryType_Enum_t.
- enum CFE_ES_AppState {
  CFE_ES_AppState_UNDEFINED = 0, CFE_ES_AppState_EARLY_INIT = 1, CFE_ES_AppState_LATE_INIT = 2, CFE_ES_AppState_RUNNING = 3,
  CFE_ES_AppState_WAITING = 4, CFE_ES_AppState_STOPPED = 5, CFE_ES_AppState_MAX }
  Label definitions associated with CFE_ES_AppState_Enum_t.

13.52.1 Typedef Documentation

13.52.1.1 CFE_ES_AppState_Enum_t

```
typedef uint32 CFE_ES_AppState_Enum_t
```

The normal progression of APP states: UNDEFINED -> EARLY_INIT -> LATE_INIT -> RUNNING -> WAITING -> STOPPED

Note

These are defined in order so that relational comparisons e.g. if (STATEA < STATEB) are possible

See also

enum CFE_ES_AppState

Definition at line 310 of file cfe_es_extern_typedefs.h.

13.52.1.2 CFE_ES_AppType_Enum_t

```
typedef uint8 CFE_ES_AppType_Enum_t
```

See also

enum CFE_ES_AppType

Definition at line 111 of file cfe_es_extern_typedefs.h.

13.52.1.3 CFE_ES_ExceptionAction_Enum_t

```
typedef uint8 CFE_ES_ExceptionAction_Enum_t
```

See also

enum CFE_ES_ExceptionAction

Definition at line 85 of file cfe_es_extern_typedefs.h.

13.52.1.4 CFE_ES_LogEntryType_Enum_t

```
typedef uint8 CFE_ES_LogEntryType_Enum_t
```

See also

enum CFE_ES_LogEntryType

Definition at line 254 of file cfe_es_extern_typedefs.h.

13.52.1.5 CFE_ES_LogMode_Enum_t

```
typedef uint8 CFE_ES_LogMode_Enum_t
```

See also

enum CFE_ES_LogMode

Definition at line 59 of file cfe_es_extern_typedefs.h.

13.52.1.6 CFE_ES_RunStatus_Enum_t

```
typedef uint32 CFE_ES_RunStatus_Enum_t
```

See also

enum CFE_ES_RunStatus

Definition at line 172 of file cfe_es_extern_typedefs.h.

13.52.1.7 CFE_ES_SystemState_Enum_t

```
typedef uint32 CFE_ES_SystemState_Enum_t
```

These values are used with the CFE_ES_WaitForSystemState() API call to synchronize application startup.

Note

These are defined in order so that relational comparisons e.g. if (STATEA $<$ STATEB) are possible

See also

enum CFE_ES_SystemState

Definition at line 227 of file cfe_es_extern_typedefs.h.

13.52.2 Enumeration Type Documentation

13.52.2.1 CFE_ES_AppState

```
enum CFE_ES_AppState
```

Enumerator

CFE_ES_AppState_UNDEFINED	Initial state before app thread is started.
CFE_ES_AppState_EARLY_INIT	App thread has started, app performing early initialization of its own data.
CFE_ES_AppState_LATE_INIT	Early/Local initialization is complete. First sync point.
CFE_ES_AppState_RUNNING	All initialization is complete. Second sync point.
CFE_ES_AppState_WAITING	Application is waiting on a Restart/Reload/Delete request.
CFE_ES_AppState_STOPPED	Application is stopped.
CFE_ES_AppState_MAX	Reserved entry, marker for the maximum state.

Definition at line 260 of file cfe_es_extern_typedefs.h.

13.52.2.2 CFE_ES_AppType

`enum CFE_ES_AppType`

Enumerator

CFE_ES_AppType_CORE	CFE core application.
CFE_ES_AppType_EXTERNAL	CFE external application.

Definition at line 91 of file cfe_es_extern_typedefs.h.

13.52.2.3 CFE_ES_ExceptionAction

`enum CFE_ES_ExceptionAction`

Enumerator

CFE_ES_ExceptionAction_RESTART_APP	Restart application if exception occurs.
CFE_ES_ExceptionAction_PROC_RESTART	Restart processor if exception occurs.

Definition at line 65 of file cfe_es_extern_typedefs.h.

13.52.2.4 CFE_ES_LogEntryType

`enum CFE_ES_LogEntryType`

Enumerator

CFE_ES_LogEntryType_CORE	Log entry from a core subsystem.
CFE_ES_LogEntryType_APPLICATION	Log entry from an application.

Definition at line 234 of file cfe_es_extern_typedefs.h.

13.52.2.5 CFE_ES_LogMode

`enum CFE_ES_LogMode`

Enumerator

CFE_ES_LogMode_OVERWRITE	Overwrite Log Mode.
CFE_ES_LogMode_DISCARD	Discard Log Mode.

Definition at line 39 of file cfe_es_extern_typedefs.h.

13.52.2.6 CFE_ES_RunStatus

`enum CFE_ES_RunStatus`

Enumerator

CFE_ES_RunStatus_APP_RUN	Indicates that the Application should continue to run.
CFE_ES_RunStatus_APP_EXIT	Indicates that the Application wants to exit normally.
CFE_ES_RunStatus_APP_ERROR	Indicates that the Application is quitting with an error.
CFE_ES_RunStatus_SYS_EXCEPTION	The cFE App caused an exception.
CFE_ES_RunStatus_SYS_RESTART	The system is requesting a restart of the cFE App.
CFE_ES_RunStatus_SYS_RELOAD	The system is requesting a reload of the cFE App.
CFE_ES_RunStatus_SYS_DELETE	The system is requesting that the cFE App is stopped.
CFE_ES_RunStatus_CORE_APP_INIT_ERROR	Indicates that the Core Application could not Init.
CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR	Indicates that the Core Application had a runtime failure.

Definition at line 117 of file cfe_es_extern_typedefs.h.

13.52.2.7 CFE_ES_SystemState

`enum CFE_ES_SystemState`

Enumerator

CFE_ES_SystemState_UNDEFINED	reserved
CFE_ES_SystemState_EARLY_INIT	single threaded mode while setting up CFE itself
CFE_ES_SystemState_CORE_STARTUP	core apps (CFE_ES_ObjectTable) are starting (multi-threaded)
CFE_ES_SystemState_CORE_READY	core is ready, starting other external apps/libraries (if any)
CFE_ES_SystemState_APPS_INIT	startup apps have all completed their early init, but not necessarily operational yet
CFE_ES_SystemState_OPERATIONAL	normal operation mode; all apps are RUNNING
CFE_ES_SystemState_SHUTDOWN	reserved for future use, all apps would be STOPPED

Definition at line 178 of file cfe_es_extern_typedefs.h.

13.53 cfe/fsw/cfe-core/src/inc/cfe_es_msg.h File Reference

```
#include "cfe.h"
#include "cfe_es.h"
```

Data Structures

- struct CFE_ES_NoArgsCmd_t

  Generic "no arguments" command.
- struct CFE_ES_RestartCmd_Payload_t

  Restart cFE Command.
- struct CFE_ES_Restart_t
- struct CFE_ES_ShellCmd_Payload_t

  Shell Command.
- struct CFE_ES_Shell_t
- struct CFE_ES_FileNameCmd_Payload_t

  Payload format for commands which accept a single file name.
- struct CFE_ES_FileNameCmd_t
- struct CFE_ES_OverWriteSysLogCmd_Payload_t

  Overwrite/Discard System Log Configuration Command.
- struct CFE_ES_OverWriteSyslog_t
- struct CFE_ES_StartAppCmd_Payload_t

  Start Application Command.
- struct CFE_ES_StartApp_t
- struct CFE_ES_AppNameCmd_Payload_t

  Command Structure for Commands requiring just an Application Name.
- struct CFE_ES_AppNameCmd_t
- struct CFE_ES_AppReloadCmd_Payload_t

  Reload Application Command.
- struct CFE_ES_ReloadApp_t
- struct CFE_ES_SetMaxPRCountCmd_Payload_t

Set Maximum Processor Reset Count Command.

- struct CFE_ES_SetMaxPRCount_t
- struct CFE_ES_DeleteCDSCmd_Payload_t

Delete Critical Data Store Command.

- struct CFE_ES_DeleteCDS_t
- struct CFE_ES_StartPerfCmd_Payload_t

Start Performance Analyzer Command.

- struct CFE_ES_StartPerfData_t
- struct CFE_ES_StopPerfCmd_Payload_t

Stop Performance Analyzer Command.

- struct CFE_ES_StopPerfData_t
- struct CFE_ES_SetPerfFilterMaskCmd_Payload_t

Set Performance Analyzer Filter Mask Command.

- struct CFE_ES_SetPerfFilterMask_t
- struct CFE_ES_SetPerfTrigMaskCmd_Payload_t

Set Performance Analyzer Trigger Mask Command.

- struct CFE_ES_SetPerfTriggerMask_t
- struct CFE_ES_SendMemPoolStatsCmd_Payload_t

Telemeter Memory Pool Statistics Command.

- struct CFE_ES_SendMemPoolStats_t
- struct CFE_ES_DumpCDSRegistryCmd_Payload_t

Dump CDS Registry Command.

- struct CFE_ES_DumpCDSRegistry_t
- struct CFE_ES_OneAppTlm_Payload_t
- struct CFE_ES_OneAppTlm_t
- struct CFE_ES_PoolStatsTlm_Payload_t
- struct CFE_ES_MemStatsTlm_t
- struct CFE_ES_HousekeepingTlm_Payload_t
- struct CFE_ES_HousekeepingTlm_t
- struct CFE_ES_ShellPacket_Payload_t
- struct CFE_ES_ShellTlm_t

Macros

Executive Services Command Codes

- #define CFE_ES_NOOP_CC 0
- #define CFE_ES_RESET_COUNTERS_CC 1
- #define CFE_ES_RESTART_CC 2
- #define CFE_ES_SHELL_CC 3
- #define CFE_ES_START_APP_CC 4
- #define CFE_ES_STOP_APP_CC 5
- #define CFE_ES_RESTART_APP_CC 6
- #define CFE_ES_RELOAD_APP_CC 7
- #define CFE_ES_QUERY_ONE_CC 8
- #define CFE_ES_QUERY_ALL_CC 9
- #define CFE_ES_CLEAR_SYSLOG_CC 10
- #define CFE_ES_WRITE_SYSLOG_CC 11
- #define CFE_ES_CLEAR_ER_LOG_CC 12
- #define CFE_ES_WRITE_ER_LOG_CC 13
- #define CFE_ES_START_PERF_DATA_CC 14

- #define CFE_ES_STOP_PERF_DATA_CC 15
- #define CFE_ES_SET_PERF_FILTER_MASK_CC 16
- #define CFE_ES_SET_PERF_TRIGGER_MASK_CC 17
- #define CFE_ES_OVER_WRITE_SYSLOG_CC 18
- #define CFE_ES_RESET_PR_COUNT_CC 19
- #define CFE_ES_SET_MAX_PR_COUNT_CC 20
- #define CFE_ES_DELETE_CDS_CC 21
- #define CFE_ES_SEND_MEM_POOL_STATS_CC 22
- #define CFE_ES_DUMP_CDS_REGISTRY_CC 23
- #define CFE_ES_QUERY_ALL_TASKS_CC 24

Typedefs

- typedef CFE_ES_NoArgsCmd_t CFE_ES_Noop_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetCounters_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearSyslog_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearERLog_t
- typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetPRCount_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAll_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllTasks_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_WriteSyslog_t
- typedef CFE_ES_FileNameCmd_t CFE_ES_WriteERLog_t
- typedef CFE_ES_AppNameCmd_t CFE_ES_StopApp_t
- typedef CFE_ES_AppNameCmd_t CFE_ES_RestartApp_t
- typedef CFE_ES_AppNameCmd_t CFE_ES_QueryOne_t
- typedef CFE_ES_HousekeepingTlm_t CFE_ES_HkPacket_t
- typedef CFE_ES_ShellTlm_t CFE_ES_ShellPacket_t
- typedef CFE_ES_MemStatsTlm_t CFE_ES_PoolStatsTlm_t

13.53.1 Macro Definition Documentation

13.53.1.1 CFE_ES_CLEAR_ER_LOG_CC

```
#define CFE_ES_CLEAR_ER_LOG_CC 12
```

Name Clears the contents of the Exeception and Reset Log

Description

This command causes the contents of the Executive Services Exception and Reset Log to be cleared.

Command Mnemonic(s) **`$sc_$cpu_ES_ClearERLog`**

Command Structure

CFE_ES_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_ERLOG1_INF_EID informational event message will be generated.
- **$sc_$cpu_ES_ERLOGINDEX -** Index into Exception Reset Log goes to zero

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not dangerous. However, any previously logged data will be lost.

See also

CFE_ES_CLEAR_SYSLOG_CC, CFE_ES_WRITE_SYSLOG_CC, CFE_ES_WRITE_ER_LOG_CC

Definition at line 602 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.2 CFE_ES_CLEAR_SYSLOG_CC

```
#define CFE_ES_CLEAR_SYSLOG_CC 10
```

Name Clear Executive Services System Log

Description

This command clears the contents of the Executive Services System Log.

Command Mnemonic(s) **$sc_$cpu_ES_ClearSysLog**

Command Structure

CFE_ES_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_SYSLOG1_INF_EID informational event message will be generated.
- **$sc_$cpu_ES_SYSLOGBYTEUSED -** System Log Bytes Used will go to zero
- **$sc_$cpu_ES_SYSLOGENTRIES -** Number of System Log Entries will go to zero

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not dangerous. However, any previously logged data will be lost.

See also

CFE_ES_WRITE_SYSLOG_CC, CFE_ES_CLEAR_ER_LOG_CC, CFE_ES_WRITE_ER_LOG_CC, CFE_ES↩_OVER_WRITE_SYSLOG_CC

Definition at line 522 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.3 CFE_ES_DELETE_CDS_CC

```
#define CFE_ES_DELETE_CDS_CC 21
```

Name Delete Critical Data Store

Description

This command allows the user to delete a Critical Data Store that was created by an Application that is now no longer executing.

Command Mnemonic(s) **$sc_$cpu_ES_DeleteCDS**

Command Structure

CFE_ES_DeleteCDS_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_CDS_DELETED_INFO_EID informational event message will be generated.
- The specified CDS should no longer appear in a CDS Registry dump generated upon receipt of the CFE_↩ ES_DUMP_CDS_REGISTRY_CC command

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified CDS is the CDS portion of a Critical Table. See CFE_TBL_DELETE_CDS_CC.
- The specified CDS is not found in the CDS Registry
- The specified CDS is associated with an Application that is still active
- An error occurred while accessing the CDS memory (see the System Log for more details)

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not critical because it is not possible to delete a CDS that is associated with an active application. However, deleting a CDS does eliminate any "history" that an application may be wishing to keep.

See also

CFE_ES_DUMP_CDS_REGISTRY_CC, CFE_TBL_DELETE_CDS_CC

Definition at line 974 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.4 CFE_ES_DUMP_CDS_REGISTRY_CC

```
#define CFE_ES_DUMP_CDS_REGISTRY_CC 23
```

Name Dump Critical Data Store Registry to a File

Description

This command allows the user to dump the Critical Data Store Registry to an onboard file.

Command Mnemonic(s) **`$sc_$cpu_ES_WriteCDS2File`**

Command Structure

CFE_ES_DumpCDSRegistry_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_CDS_REG_DUMP_INF_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE configuration parameter) will be updated with the lastest information.

Error Conditions

This command may fail for the following reason(s):

- Error occurred while trying to create the dump file

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_ES_DELETE_CDS_CC, CFE_TBL_DELETE_CDS_CC

Definition at line 1057 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.5 CFE_ES_NOOP_CC

```
#define CFE_ES_NOOP_CC 0
```

Name Executive Services No-Op

Description

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Executive Services task.

Command Mnemonic(s) **`$sc_$cpu_ES_NOOP`**

Command Structure

CFE_ES_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_NOOP_INF_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- the CFE_ES_LEN_ERR_EID error event message will be generated

Criticality

None

See also

Definition at line 82 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.6 CFE_ES_OVER_WRITE_SYSLOG_CC

```
#define CFE_ES_OVER_WRITE_SYSLOG_CC 18
```

Name Set Executive Services System Log Mode to Discard/Overwrite

Description

This command allows the user to configure the Executive Services to either discard new System Log messages when it is full or to overwrite the oldest messages.

Command Mnemonic(s) **`$sc_$cpu_ES_OverwriteSysLogMode`**

Command Structure

CFE_ES_OverWriteSyslog_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_ES_SYSLOGMODE`** - Current System Log Mode should reflect the commanded value
- The CFE_ES_SYSLOGMODE_EID debug event message will be generated.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The desired mode is neither CFE_ES_LogMode_OVERWRITE or CFE_ES_LogMode_DISCARD

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

None. (It should be noted that "Overwrite" mode would allow a message identifying the cause of a problem to be lost by a subsequent flood of additional messages).

See also

CFE_ES_CLEAR_SYSLOG_CC, CFE_ES_WRITE_SYSLOG_CC

Definition at line 850 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.7 CFE_ES_QUERY_ALL_CC

```
#define CFE_ES_QUERY_ALL_CC 9
```

Name Writes all Executive Services Information on All Applications to a File

Description

This command takes the information kept by Executive Services on all of the registered applications and writes it to the specified file.

Command Mnemonic(s) **$sc_$cpu_ES_WriteAppInfo2File**

Command Structure

CFE_ES_FileNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC** - command execution counter will increment
- The CFE_ES_ALL_APPS_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_APP_↩ LOG_FILE configuration parameter) will be updated with the lastest information.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_ES_QUERY_ONE_CC, CFE_ES_QUERY_ALL_TASKS_CC

Definition at line 484 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.8 CFE_ES_QUERY_ALL_TASKS_CC

```
#define CFE_ES_QUERY_ALL_TASKS_CC 24
```

Name Writes a list of All Executive Services Tasks to a File

Description

This command takes the information kept by Executive Services on all of the registered tasks and writes it to the specified file.

Command Mnemonic(s) **$sc_$cpu_ES_WriteTaskInfo2File**

Command Structure

CFE_ES_FileNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_TASKINFO_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_TASK↩ _LOG_FILE configuration parameter) will be updated with the lastest information.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_ES_QUERY_ALL_CC, CFE_ES_QUERY_ONE_CC

Definition at line 1099 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.9 CFE_ES_QUERY_ONE_CC

```
#define CFE_ES_QUERY_ONE_CC 8
```

Name Request Executive Services Information on a Specified Application

Description

This command takes the information kept by Executive Services on the specified application and telemeters it to the ground.

Command Mnemonic(s) **$sc_$cpu_ES_QueryApp**

Command Structure

CFE_ES_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_ONE_APP_EID debug event message will be generated. NOTE: This event message only identifies that the act of stopping the application has begun, not that is has completed.
- Receipt of the CFE_ES_OneAppTlm_t telemetry packet

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified application name is not recognized as an active application

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

None

See also

CFE_ES_QUERY_ALL_CC, CFE_ES_QUERY_ALL_TASKS_CC

Definition at line 442 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.10 CFE_ES_RELOAD_APP_CC

```
#define CFE_ES_RELOAD_APP_CC 7
```

Name Stops, Unloads, Loads from a File and Restarts an Application

Description

This command halts and removes the specified Application from the system. Then it immediately loads the Application from the command specified file and restarts it. This command is especially useful for restarting a Command Ingest Application since once it has been stopped, no further commands can come in to restart it.

Command Mnemonic(s) **`$sc_$cpu_ES_ReloadApp`**

Command Structure

CFE_ES_ReloadApp_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_RELOAD_APP_DBG_EID debug event message will be generated. NOTE: This event message only identifies that the act of stopping the application has begun, not that is has completed.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified application name is not recognized as an active application
- The specified application is one of the cFE's Core applications (ES, EVS, SB, TBL, TIME)

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases
- Additional information on the reason for command failure may be found in the System Log

Criticality

This command is not inherently dangerous, however the restarting of certain applications (e.g. - Spacecraft Attitude and Control) may have a detrimental effect on the spacecraft.

See also

CFE_ES_START_APP_CC, CFE_ES_STOP_APP_CC, CFE_ES_RESTART_APP_CC

Definition at line 404 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.11 CFE_ES_RESET_COUNTERS_CC

```
#define CFE_ES_RESET_COUNTERS_CC 1
```

Name Executive Services Reset Counters

Description

This command resets the following counters within the Executive Services housekeeping telemetry:

- Command Execution Counter
- Command Error Counter

Command Mnemonic(s) **`$sc_$cpu_ES_ResetCtrs`**

Command Structure

CFE_ES_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_RESET_INF_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- the CFE_ES_LEN_ERR_EID error event message will be generated

Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

See also

CFE_ES_RESET_PR_COUNT_CC

Definition at line 121 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.12 CFE_ES_RESET_PR_COUNT_CC

```
#define CFE_ES_RESET_PR_COUNT_CC 19
```

Name Resets the Processor Reset Counter to Zero

Description

This command allows the user to reset the Processor Reset Counter to zero. The Processor Reset Counter counts the number of Processor Resets that have occurred so as to identify when a Processor Reset should automatically be upgraded to a full Power-On Reset.

Command Mnemonic(s) **`$sc_$cpu_ES_ResetPRCnt`**

Command Structure

CFE_ES_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_ES_ProcResetCnt`** - Current number of processor resets will go to zero
- The CFE_ES_RESET_PR_COUNT_EID informational event message will be generated.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not critical. The only impact would be that the system would have to have more processor resets before an automatic power-on reset occurred.

See also

CFE_ES_SET_MAX_PR_COUNT_CC, CFE_ES_RESET_COUNTERS_CC

Definition at line 890 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.13 CFE_ES_RESTART_APP_CC

```
#define CFE_ES_RESTART_APP_CC 6
```

Name Stops and Restarts an Application

Description

This command halts and restarts the specified Application. This command does **NOT** reload the application from the onboard filesystem.

Command Mnemonic(s) **`$sc_$cpu_ES_ResetApp`**

Command Structure

CFE_ES_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_RESTART_APP_DBG_EID debug event message will be generated. NOTE: This event message only identifies that the act of stopping the application has begun, not that is has completed.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified application name is not recognized as an active application
- The specified application is one of the cFE's Core applications (ES, EVS, SB, TBL, TIME)

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases
- Additional information on the reason for command failure may be found in the System Log

Criticality

This command is not inherently dangerous, however the restarting of certain applications (e.g. - Spacecraft Attitude and Control) may have a detrimental effect on the spacecraft.

See also

CFE_ES_START_APP_CC, CFE_ES_STOP_APP_CC, CFE_ES_RELOAD_APP_CC

Definition at line 358 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.14 CFE_ES_RESTART_CC

```
#define CFE_ES_RESTART_CC 2
```

Name Executive Services Processor / Power-On Reset

Description

This command restarts the cFE in one of two modes. The Power-On Reset will cause the cFE to restart as though the power were first applied to the processor. The Processor Reset will attempt to retain the contents of the volatile disk and the contents of the Critical Data Store. NOTE: If a requested Processor Reset should cause the Processor Reset Counter (**$sc_$cpu_ES_ProcResetCnt)** to exceed OR EQUAL the limit CFE_PLATFORM_ES_MAX↩ _PROCESSOR_RESETS (which is reported in housekeeping telemetry as **$sc_$cpu_ES_MaxProcResets)**, the command is **AUTOMATICALLY** upgraded to a Power-On Reset.

Command Mnemonic(s) **$sc_$cpu_ES_ProcessorReset**, **$sc_$cpu_ES_PowerOnReset**

Command Structure

CFE_ES_RestartCmd_Payload_t

Command Verification

Successful execution of this command (as a Processor Reset) may be verified with the following telemetry:

- **$sc_$cpu_ES_ProcResetCnt -** processor reset counter will increment
- New entries in the Exception Reset Log and System Log can be found
  NOTE: Verification of a Power-On Reset is shown through the loss of data nominally retained through a Processor Reset
  NOTE: Since the reset of the processor resets the command execution counter (**$sc_$cpu_ES_CMDPC)**, this counter **CANNOT** be used to verify command execution.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The Restart Type was not a recognized value.

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- the CFE_ES_BOOT_ERR_EID error event message will be generated

Criticality

This command is, by definition, dangerous. Significant loss of data will occur. All processes and the cFE itself will be stopped and restarted. With the Power-On reset option, all data on the volatile disk and the contents of the Critical Data Store will be lost.

See also

CFE_ES_RESET_PR_COUNT_CC, CFE_ES_SET_MAX_PR_COUNT_CC

Definition at line 171 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.15 CFE_ES_SEND_MEM_POOL_STATS_CC

```
#define CFE_ES_SEND_MEM_POOL_STATS_CC 22
```

Name Telemeter Memory Pool Statistics

Description

This command allows the user to obtain a snapshot of the statistics maintained for a specified memory pool.

Command Mnemonic(s) **`$sc_$cpu_ES_PoolStats`**

Command Structure

CFE_ES_SendMemPoolStats_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_TLM_POOL_STATS_INFO_EID debug event message will be generated.
- The Memory Pool Statistics Telemetry Packet is produced

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified handle is not associated with a known memory pool
- The specified handle caused a processor exception because it improperly identified a segment of memory

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

An incorrect Memory Pool Handle value can cause a system crash. Extreme care should be taken to ensure the memory handle value used in the command is correct.

See also

Definition at line 1016 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.16 CFE_ES_SET_MAX_PR_COUNT_CC

```
#define CFE_ES_SET_MAX_PR_COUNT_CC 20
```

Name Configure the Maximum Number of Processor Resets before a Power-On Reset

Description

This command allows the user to specify the number of Processor Resets that are allowed before the next Processor Reset is upgraded to a Power-On Reset.

Command Mnemonic(s) **`$sc_$cpu_ES_SetMaxPRCnt`**

Command Structure

CFE_ES_SetMaxPRCount_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_ES_MaxProcResets`** - Current maximum number of processor resets before an automatic power-on reset will go to the command specified value.
- The CFE_ES_SET_MAX_PR_COUNT_EID informational event message will be generated.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

If the operator were to set the Maximum Processor Reset Count to too high a value, the processor would require an inordinate number of consecutive processor resets before an automatic power-on reset would occur. This could potentially leave the spacecraft without any control for a significant amount of time if a processor reset fails to clear a problem.

See also

CFE_ES_RESET_PR_COUNT_CC

Definition at line 931 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.17 CFE_ES_SET_PERF_FILTER_MASK_CC

```
#define CFE_ES_SET_PERF_FILTER_MASK_CC 16
```

Name Set Performance Analyzer's Filter Masks

Description

This command sets the Performance Analyzer's Filter Masks.

Command Mnemonic(s) **$sc_$cpu_ES_LAFilterMask**

Command Structure

CFE_ES_SetPerfFilterMask_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- **$sc_$cpu_ES_PerfFltrMask[MaskCnt**] - the current performance filter mask value(s) should reflect the commanded value
- The CFE_ES_PERF_FILTMSKCMD_EID debug event message will be generated.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The Filter Mask ID number is out of range

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

Changing the filter masks may cause a small change in the Performance Analyzer's CPU utilization.

See also

CFE_ES_START_PERF_DATA_CC, CFE_ES_STOP_PERF_DATA_CC, CFE_ES_SET_PERF_TRIGGER_M↩ASK_CC

Definition at line 771 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.18 CFE_ES_SET_PERF_TRIGGER_MASK_CC

```
#define CFE_ES_SET_PERF_TRIGGER_MASK_CC 17
```

Name Set Performance Analyzer's Trigger Masks

Description

This command sets the Performance Analyzer's Trigger Masks.

Command Mnemonic(s) **`$sc_$cpu_ES_LATriggerMask`**

Command Structure

CFE_ES_SetPerfTriggerMask_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_ES_PerfTrigMask[MaskCnt]`** - the current performance trigger mask value(s) should reflect the commanded value
- The CFE_ES_PERF_TRIGMSKCMD_EID debug event message will be generated.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The Trigger Mask ID number is out of range

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

Changing the trigger masks may cause a small change in the Performance Analyzer's CPU utilization.

See also

CFE_ES_START_PERF_DATA_CC, CFE_ES_STOP_PERF_DATA_CC, CFE_ES_SET_PERF_FILTER_MASK_CC

Definition at line 809 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.19 CFE_ES_SHELL_CC

```
#define CFE_ES_SHELL_CC 3
```

Name Executive Services O/S Shell Command

Description

This command passes an ASCII string as a command line to the underlying realtime operating system shell. Any response to the command is both written to the shell command output file and sent as a series of shell command output telemetry packets.

If the shell command output filename argument is empty, then CFE_PLATFORM_ES_DEFAULT_SHELL_FILENAME will be used as the filename.

Command Mnemonic(s) **$sc_$cpu$_ES_Shell**

Command Structure

CFE_ES_Shell_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_SHELL_INF_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- Failure to create the shell command output file
- The shell command started with ES_ but was not one of the recognized cFE shell commands
- There was an error while performing a OS_lseek on the shell command output file
- There was an error while redirecting the shell command response to the shell command output file

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- the CFE_ES_SHELL_ERR_EID error event message will be generated
- Additional information on the error should be found in the System Log

Criticality

This command should be used with caution. Interfering with the operation of the underlying realtime operating system can cause significant problems.

See also

Definition at line 219 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.20 CFE_ES_START_APP_CC

```
#define CFE_ES_START_APP_CC 4
```

Name Load and Start an Application

Description

This command starts the specified application with the specified start address, stack size, etc options.

Command Mnemonic(s) **$sc_$cpu_ES_StartApp**

Command Structure

CFE_ES_StartApp_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_START_INF_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified application filename string is either a NULL string or less than four characters in length
- The specified application entry point is a NULL string
- The specified application name is a NULL string
- The specified stack size is less than CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
- The specified priority is greater than MAX_PRIORITY (as defined in osapi.c)
- The specified exception action is neither CFE_ES_ExceptionAction_RESTART_APP (0) or CFE_ES_↩ ExceptionAction_PROC_RESTART (1)
- The Operating System was unable to load the specified application file

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous although system resources could be taxed beyond their limits with the starting of erroneous or invalid applications.

See also

CFE_ES_STOP_APP_CC, CFE_ES_RESTART_APP_CC, CFE_ES_RELOAD_APP_CC

Definition at line 265 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.21 CFE_ES_START_PERF_DATA_CC

```
#define CFE_ES_START_PERF_DATA_CC 14
```

Name Start Performance Analyzer

Description

This command causes the Performance Analyzer to begin collecting data using the specified trigger mode.

Command Mnemonic(s) **$sc_$cpu_ES_StartLAData**

Command Structure

CFE_ES_StartPerfData_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- **$sc_$cpu_ES_PerfState -** Current performance analyzer state will change to either WAITING FOR TRIGGER or, if conditions are appropriate fast enough, TRIGGERED.
- **$sc_$cpu_ES_PerfMode -** Performance Analyzer Mode will change to the commanded trigger mode (TRIGGER START, TRIGGER CENTER, or TRIGGER END).
- **$sc_$cpu_ES_PerfTrigCnt -** Performance Trigger Count will go to zero
- **$sc_$cpu_ES_PerfDataStart -** Data Start Index will go to zero
- **$sc_$cpu_ES_PerfDataEnd -** Data End Index will go to zero
- **$sc_$cpu_ES_PerfDataCnt -** Performance Data Counter will go to zero
- The CFE_ES_PERF_STARTCMD_EID debug event message will be generated.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- A previous CFE_ES_STOP_PERF_DATA_CC command has not completely finished.
- An invalid trigger mode is requested.

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous but may cause a small increase in CPU utilization as the performance analyzer data is collected.

See also

CFE_ES_STOP_PERF_DATA_CC, CFE_ES_SET_PERF_FILTER_MASK_CC, CFE_ES_SET_PERF_TRIGG↩ ER_MASK_CC

Definition at line 690 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.22 CFE_ES_STOP_APP_CC

```
#define CFE_ES_STOP_APP_CC 5
```

Name Stop and Unload Application

Description

This command halts and removes the specified Application from the system. **NOTE:** This command should never be used on the Command Ingest application. This would prevent further commands from entering the system. If Command Ingest needs to be stopped and restarted, use CFE_ES_RESTART_APP_CC or CFE_ES_RELOAD_APP_CC.

Command Mnemonic(s) **$sc_$cpu_ES_StopApp**

Command Structure

CFE_ES_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_ES_CMDPC -** command execution counter will increment
- The CFE_ES_STOP_DBG_EID debug event message will be generated. NOTE: This event message only identifies that the stop has been started, not that is has completed.
- Once the stop has successfully completed, the list of Applications and Tasks created in response to the **$sc_$cpu_ES_WriteAppInfo2File**, **$sc_$cpu_ES_WriteTaskInfo2File** **should** no longer contain the specified application.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- The specified application name is not recognized as an active application
- The specified application is one of the cFE's Core applications (ES, EVS, SB, TBL, TIME)

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_ES_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases
- Additional information on the reason for command failure may be found in the System Log

Criticality

This command is not inherently dangerous, however the removal of certain applications (e.g. - Spacecraft Attitude and Control) may have a detrimental effect on the spacecraft.

See also

CFE_ES_START_APP_CC, CFE_ES_RESTART_APP_CC, CFE_ES_RELOAD_APP_CC

Definition at line 315 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.23 CFE_ES_STOP_PERF_DATA_CC

```
#define CFE_ES_STOP_PERF_DATA_CC 15
```

Name Stop Performance Analyzer

Description

This command stops the Performance Analyzer from collecting any more data.

Command Mnemonic(s) **`$sc_$cpu_ES_StopLAData`**

Command Structure

CFE_ES_StopPerfData_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_ES_PerfState`** - Current performance analyzer state will change to IDLE.
- The CFE_ES_PERF_STOPCMD_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME configuration parameter) will be updated with the lastest information.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- A previous Stop Performance Analyzer command is still in process
- An error occurred while spawning the child task responsible for dumping the Performance Analyzer data to a file

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. An additional low priority child task will be spawned, however, to dump the performance analyzer data to a file.

See also

CFE_ES_START_PERF_DATA_CC, CFE_ES_SET_PERF_FILTER_MASK_CC, CFE_ES_SET_PERF_TRIGGER_MASK_CC

Definition at line 733 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.24 CFE_ES_WRITE_ER_LOG_CC

```
#define CFE_ES_WRITE_ER_LOG_CC 13
```

Name Writes Exeception and Reset Log to a File

Description

This command causes the contents of the Executive Services Exception and Reset Log to be written to the specified file.

Command Mnemonic(s) **`$sc_$cpu_ES_WriteERLog2File`**

Command Structure

CFE_ES_FileNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_ERLOG2_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_ER_L↩OG_FILE configuration parameter) will be updated with the lastest information.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_ES_CLEAR_SYSLOG_CC, CFE_ES_WRITE_SYSLOG_CC, CFE_ES_CLEAR_ER_LOG_CC

Definition at line 644 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.1.25 CFE_ES_WRITE_SYSLOG_CC

```
#define CFE_ES_WRITE_SYSLOG_CC 11
```

Name Writes contents of Executive Services System Log to a File

Description

This command causes the contents of the Executive Services System Log to be written to a log file.

Command Mnemonic(s) **`$sc_$cpu_ES_WriteSysLog2File`**

Command Structure

CFE_ES_FileNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_ES_CMDPC`** - command execution counter will increment
- The CFE_ES_SYSLOG2_EID debug event message will be generated.
- The file specified in the command (or the default specified by the CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE configuration parameter) will be updated with the lastest information.

Error Conditions

This command may fail for the following reason(s):

- The command packet length is incorrect
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_ES_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_ES_CLEAR_SYSLOG_CC, CFE_ES_CLEAR_ER_LOG_CC, CFE_ES_WRITE_ER_LOG_CC, CFE_ES_OVER_WRITE_SYSLOG_CC

Definition at line 565 of file cfe_es_msg.h.

Referenced by CFE_ES_TaskPipe().

13.53.2 Typedef Documentation

13.53.2.1 CFE_ES_ClearERLog_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearERLog_t
```

Definition at line 1134 of file cfe_es_msg.h.

13.53.2.2 CFE_ES_ClearSyslog_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearSyslog_t
```

Definition at line 1133 of file cfe_es_msg.h.

13.53.2.3 CFE_ES_HkPacket_t

```
typedef CFE_ES_HousekeepingTlm_t CFE_ES_HkPacket_t
```

Definition at line 1604 of file cfe_es_msg.h.

13.53.2.4 CFE_ES_Noop_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_Noop_t
```

Definition at line 1131 of file cfe_es_msg.h.

13.53.2.5 CFE_ES_PoolStatsTlm_t

```
typedef CFE_ES_MemStatsTlm_t CFE_ES_PoolStatsTlm_t
```

Definition at line 1606 of file cfe_es_msg.h.

13.53.2.6 CFE_ES_QueryAll_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAll_t
```

Definition at line 1199 of file cfe_es_msg.h.

13.53.2.7 CFE_ES_QueryAllTasks_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllTasks_t
```

Definition at line 1200 of file cfe_es_msg.h.

13.53.2.8 CFE_ES_QueryOne_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_QueryOne_t
```

Definition at line 1276 of file cfe_es_msg.h.

13.53.2.9 CFE_ES_ResetCounters_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetCounters_t
```

Definition at line 1132 of file cfe_es_msg.h.

13.53.2.10 CFE_ES_ResetPRCount_t

```
typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetPRCount_t
```

Definition at line 1135 of file cfe_es_msg.h.

13.53.2.11 CFE_ES_RestartApp_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_RestartApp_t
```

Definition at line 1275 of file cfe_es_msg.h.

13.53.2.12 CFE_ES_ShellPacket_t

```
typedef CFE_ES_ShellTlm_t CFE_ES_ShellPacket_t
```

Definition at line 1605 of file cfe_es_msg.h.

13.53.2.13 CFE_ES_StopApp_t

```
typedef CFE_ES_AppNameCmd_t CFE_ES_StopApp_t
```

Definition at line 1274 of file cfe_es_msg.h.

13.53.2.14 CFE_ES_WriteERLog_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_WriteERLog_t
```

Definition at line 1202 of file cfe_es_msg.h.

13.53.2.15 CFE_ES_WriteSyslog_t

```
typedef CFE_ES_FileNameCmd_t CFE_ES_WriteSyslog_t
```

Definition at line 1201 of file cfe_es_msg.h.

13.54 cfe/fsw/cfe-core/src/inc/cfe_evs.h File Reference

```
#include "cfe_evs_extern_typedefs.h"
#include "common_types.h"
#include "cfe_time.h"
#include "cfe_evs_msg.h"
#include "osapi.h"
#include "cfe_sb.h"
```

Data Structures

- struct CFE_EVS_BinFilter_t

Macros

- #define CFE_EVS_BINARY_FILTER CFE_EVS_EventFilter_BINARY
- #define CFE_EVS_PORT1 CFE_EVS_EventOutput_PORT1
- #define CFE_EVS_PORT2 CFE_EVS_EventOutput_PORT2
- #define CFE_EVS_PORT3 CFE_EVS_EventOutput_PORT3
- #define CFE_EVS_PORT4 CFE_EVS_EventOutput_PORT4
- #define CFE_EVS_DEBUG CFE_EVS_EventType_DEBUG
- #define CFE_EVS_INFORMATION CFE_EVS_EventType_INFORMATION
- #define CFE_EVS_ERROR CFE_EVS_EventType_ERROR
- #define CFE_EVS_CRITICAL CFE_EVS_EventType_CRITICAL

Common Event Filter Mask Values

- #define CFE_EVS_NO_FILTER 0x0000
  Stops any filtering. All messages are sent.
- #define CFE_EVS_FIRST_ONE_STOP 0xFFFF
  Sends the first event. All remaining messages are filtered.
- #define CFE_EVS_FIRST_TWO_STOP 0xFFFE
  Sends the first 2 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_4_STOP 0xFFFC
  Sends the first 4 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_8_STOP 0xFFF8
  Sends the first 8 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_16_STOP 0xFFF0
  Sends the first 16 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_32_STOP 0xFFE0
  Sends the first 32 events. All remaining messages are filtered.
- #define CFE_EVS_FIRST_64_STOP 0xFFC0
  Sends the first 64 events. All remaining messages are filtered.
- #define CFE_EVS_EVERY_OTHER_ONE 0x0001
  Sends every other event.
- #define CFE_EVS_EVERY_OTHER_TWO 0x0002
  Sends two, filters one, sends two, filters one, etc.
- #define CFE_EVS_EVERY_FOURTH_ONE 0x0003
  Sends every fourth event message. All others are filtered.

Functions

- int32 CFE_EVS_Register (void *Filters, uint16 NumFilteredEvents, uint16 FilterScheme)
  Register an application for receiving event services.
- int32 CFE_EVS_Unregister (void)
  Cleanup internal structures used by the event manager for the calling Application.
- int32 CFE_EVS_SendEvent (uint16 EventID, uint16 EventType, const char *Spec,...) OS_PRINTF(3
  Generate a software event.
- int32 int32 CFE_EVS_SendEventWithAppID (uint16 EventID, uint16 EventType, uint32 AppID, const char *Spec,...) OS_PRINTF(4
  Generate a software event given the specified Application ID.
- int32 int32 int32 CFE_EVS_SendTimedEvent (CFE_TIME_SysTime_t Time, uint16 EventID, uint16 EventType, const char *Spec,...) OS_PRINTF(4
  Generate a software event with a specific time tag.
- int32 int32 int32 int32 CFE_EVS_ResetFilter (int16 EventID)
  Resets the calling application's event filter for a single event ID.
- int32 CFE_EVS_ResetAllFilters (void)
  Resets all of the calling application's event filters.

13.54.1 Macro Definition Documentation

13.54.1.1 CFE_EVS_BINARY_FILTER

```
#define CFE_EVS_BINARY_FILTER CFE_EVS_EventFilter_BINARY
```

Definition at line 88 of file cfe_evs.h.

13.54.1.2 CFE_EVS_CRITICAL

```
#define CFE_EVS_CRITICAL CFE_EVS_EventType_CRITICAL
```

Definition at line 104 of file cfe_evs.h.

13.54.1.3 CFE_EVS_DEBUG

```
#define CFE_EVS_DEBUG CFE_EVS_EventType_DEBUG
```

Definition at line 101 of file cfe_evs.h.

13.54.1.4 CFE_EVS_ERROR

```
#define CFE_EVS_ERROR CFE_EVS_EventType_ERROR
```

Definition at line 103 of file cfe_evs.h.

13.54.1.5 CFE_EVS_EVERY_FOURTH_ONE

```
#define CFE_EVS_EVERY_FOURTH_ONE 0x0003
```

Definition at line 74 of file cfe_evs.h.

13.54.1.6 CFE_EVS_EVERY_OTHER_ONE

```
#define CFE_EVS_EVERY_OTHER_ONE 0x0001
```

Definition at line 72 of file cfe_evs.h.

13.54.1.7 CFE_EVS_EVERY_OTHER_TWO

```
#define CFE_EVS_EVERY_OTHER_TWO 0x0002
```

Definition at line 73 of file cfe_evs.h.

13.54.1.8 CFE_EVS_FIRST_16_STOP

```
#define CFE_EVS_FIRST_16_STOP 0xFFF0
```

Definition at line 69 of file cfe_evs.h.

13.54.1.9 CFE_EVS_FIRST_32_STOP

```
#define CFE_EVS_FIRST_32_STOP 0xFFE0
```

Definition at line 70 of file cfe_evs.h.

13.54.1.10 CFE_EVS_FIRST_4_STOP

```
#define CFE_EVS_FIRST_4_STOP 0xFFFC
```

Definition at line 67 of file cfe_evs.h.

13.54.1.11 CFE_EVS_FIRST_64_STOP

```
#define CFE_EVS_FIRST_64_STOP 0xFFC0
```

Definition at line 71 of file cfe_evs.h.

13.54.1.12 CFE_EVS_FIRST_8_STOP

```
#define CFE_EVS_FIRST_8_STOP 0xFFF8
```

Definition at line 68 of file cfe_evs.h.

13.54.1.13 CFE_EVS_FIRST_ONE_STOP

```
#define CFE_EVS_FIRST_ONE_STOP 0xFFFF
```

Definition at line 65 of file cfe_evs.h.

13.54.1.14 CFE_EVS_FIRST_TWO_STOP

```
#define CFE_EVS_FIRST_TWO_STOP 0xFFFE
```

Definition at line 66 of file cfe_evs.h.

13.54.1.15 CFE_EVS_INFORMATION

```
#define CFE_EVS_INFORMATION CFE_EVS_EventType_INFORMATION
```

Definition at line 102 of file cfe_evs.h.

13.54.1.16 CFE_EVS_NO_FILTER

```
#define CFE_EVS_NO_FILTER 0x0000
```

Definition at line 64 of file cfe_evs.h.

13.54.1.17 CFE_EVS_PORT1

```
#define CFE_EVS_PORT1 CFE_EVS_EventOutput_PORT1
```

Definition at line 93 of file cfe_evs.h.

13.54.1.18 CFE_EVS_PORT2

```
#define CFE_EVS_PORT2 CFE_EVS_EventOutput_PORT2
```

Definition at line 94 of file cfe_evs.h.

13.54.1.19 CFE_EVS_PORT3

```
#define CFE_EVS_PORT3 CFE_EVS_EventOutput_PORT3
```

Definition at line 95 of file cfe_evs.h.

13.54.1.20 CFE_EVS_PORT4

```
#define CFE_EVS_PORT4 CFE_EVS_EventOutput_PORT4
```

Definition at line 96 of file cfe_evs.h.

13.54.2 Function Documentation

13.54.2.1 CFE_EVS_Register()

```
int32 CFE_EVS_Register (
            void * Filters,
            uint16 NumFilteredEvents,
            uint16 FilterScheme )
```

Description

This routine registers an application with event services and allocates/initializes the internal data structures used to support this application's events. An application may not send events unless it has called this routine. The routine also accepts a filter array structure for applications requiring event filtering. In the current implementation of the EVS, only the binary filtering scheme is supported. See section TBD of the cFE Application Programmer's Guide for a description of the behavior of binary filters. Applications may call CFE_EVS_Register more than once, but each call will wipe out all filters registered by previous calls (filter registration is NOT cumulative).

Assumptions, External Events, and Notes:

Note: Event filters can be added, deleted or modified by ground commands. All filtering schemes include a default setting that results in no filtering (such as CFE_EVS_NO_FILTER for binary filters).

Filter Scheme: Binary
Code: CFE_EVS_EventFilter_BINARY
Filter Structure:

```
typedef struct {
      uint16  EventID,
      uint16  Mask ;
} CFE_EVS_BinFilter_t;
```

Parameters

in	*Filters*	Pointer to an array of event message filters, or NULL if no filtering is desired. The structure of an event message filter depends on the FilterScheme selected. (see Filter Schemes mentioned above)
in	*NumFilteredEvents*	The number of event message filters included in this call. This must be less than or equal to the maximum number of events allowed per application (CFE_PLATFORM_EVS_MAX_EVENT_FILTERS).
in	*FilterScheme*	The event filtering scheme that this application will use. For the first implementation of the event services, only filter type CFE_EVS_EventFilter_BINARY will be supported.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_FILTER_OVERLOAD	Number of Application event filters input upon registration is greater than CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
CFE_EVS_UNKNOWN_FILTER	CFE_EVS_Register() FilterScheme parameter was illegal
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	

Returns

See also

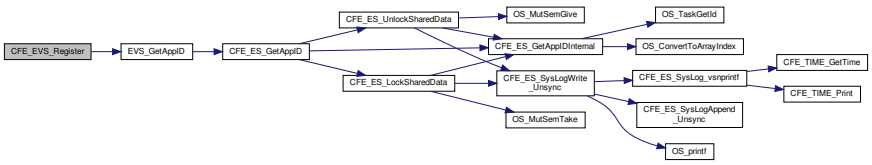
CFE_EVS_Unregister

Definition at line 60 of file cfe_evs.c.

References EVS_AppData_t::ActiveFlag, CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_ES_ERR_BUFFER, CFE_EVS_EventFilter_BINARY, CFE_EVS_FREE_SLOT, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_EVS_UNKNOWN_FILTER, CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SUCCESS, EVS_BinFilter_t::Count, EVS_AppData_t::EventCount, EVS_BinFilter_t::EventID, CFE_EVS_BinFilter_t::EventID, EVS_AppData_t::EventTypesActiveFlag, EVS_GetAppID(), EVS_BinFilter_t::Mask, CFE_EVS_BinFilter_t::Mask, NULL, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), and CFE_SB_AppInit().

Here is the call graph for this function:



13.54.2.2 CFE_EVS_ResetAllFilters()

```
int32 CFE_EVS_ResetAllFilters (
            void  )
```

Description

This routine resets all the calling application's event filter counters to zero, providing a quick and convenient method for resetting event filters.

Assumptions, External Events, and Notes:

None

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	

Returns

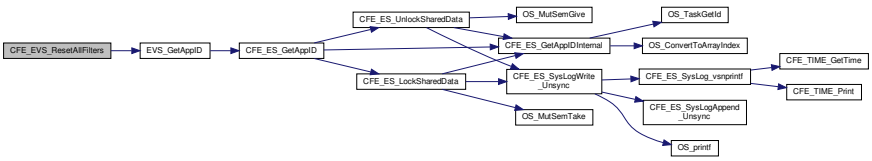
See also

CFE_EVS_ResetFilter

Definition at line 349 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_SUCC↩ESS, EVS_BinFilter_t::Count, EVS_GetAppID(), and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



13.54.2.3 CFE_EVS_ResetFilter()

```
int32 int32 int32 int32 CFE_EVS_ResetFilter (
            int16 EventID )
```

Description

The effect of resetting an event filter depends on the filter scheme. The CFE_EVS_EventFilter_BINARY scheme resets the filter counter for the specified Event ID.

Assumptions, External Events, and Notes:

None

Parameters

in	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	

Returns

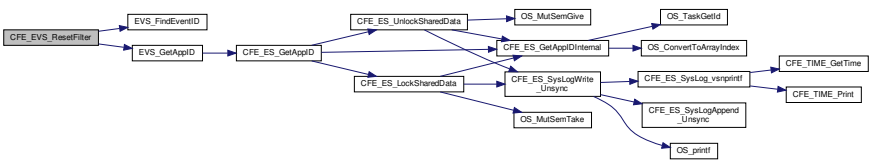
See also

CFE_EVS_ResetAllFilters

Definition at line 304 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, EVS_AppData_t::BinFilters, CFE_EVS_APP_NOT_REGISTERED, CFE_EVS_EVT_NOT_REGISTERED, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_SUCCESS, EVS_↩ BinFilter_t::Count, EVS_FindEventID(), EVS_GetAppID(), NULL, and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



13.54.2.4 CFE_EVS_SendEvent()

```
int32 CFE_EVS_SendEvent (
            uint16 EventID,
            uint16 EventType,
            const char * Spec,
             ... )
```

Description

This routine generates a software event message. If the EventID is not filtered, the event will be sent as a software bus message, optionally logged in the local event log, and optionally sent as an ASCII text string out the enabled output port(s).

Assumptions, External Events, and Notes:

None

Parameters

in	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.
in	*EventType*	A numeric literal used to classify an event, one of: • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	*Spec*	A pointer to a null terminated text string describing the output format for the event. This is the same type of format string used for the ANSI `printf` function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter CFE_MISSION_EVS_MAX_MESSAGE_LENGTH. Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	
Any of the error codes from CFE_SB_SendMsg	

Returns

See also

CFE_EVS_SendEventWithAppID, CFE_EVS_SendTimedEvent

Referenced by CFE_ES_ClearERLogCmd(), CFE_ES_ClearSyslogCmd(), CFE_ES_DeleteCDSCmd(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_FileWriteByteCntErr(), CFE_ES_NoopCmd(), CFE_ES_OverWriteSyslogCmd(), CFE_ES_PerfLogDump(), CFE_ES_ProcessControlRequest(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_ResetCountersCmd(), CFE_ES_ResetPRCountCmd(), CFE_ES_RestartAppCmd(), CFE_ES_RestartCmd(), CFE_ES_SendMemPoolStatsCmd(), CFE_ES_SetMaxPRCountCmd(), CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), CFE_ES_ShellCmd(), CFE_ES_StartAppCmd(), CFE_ES_StartPerfDataCmd(), CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_SysLogDump(), CFE_ES_TaskInit(), CFE_ES_TaskPipe(), CFE_ES_VerifyCmdLength(), CFE_SB_AppInit(), CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_FileWriteByteCntErr(), CFE_SB_NoopCmd(), CFE_SB_ProcessCmdPipePkt(), CFE_SB_ResetCountersCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRtgInfo(), CFE_SB_SendStatsCmd(), and CFE_SB_VerifyCmdLength().

13.54.2.5 CFE_EVS_SendEventWithAppID()

```
int32 int32 CFE_EVS_SendEventWithAppID (
            uint16 EventID,
            uint16 EventType,
            uint32 AppID,
            const char * Spec,
             ...  )
```

Description

This routine generates a software event message. If the EventID is not filtered, the event will be sent as a software bus message, optionally logged in the local event log, and optionally sent as an ASCII text string out the enabled output port(s). Note that this function should really only be used from within an API in order to preserve the context of an Application's event. In general, CFE_EVS_SendEvent should be used.

Assumptions, External Events, and Notes:

None

Parameters

in	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.
in	*EventType*	A numeric literal used to classify an event, one of: • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	*AppID*	The Application ID from which the event message should appear.

Parameters

in	*Spec*	A pointer to a null terminated text string describing the output format for the event. This is the same type of format string used for the ANSI `printf` function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter CFE_MISSION_EVS_MAX_MESSAGE_LENGTH. Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	
Any of the error codes from CFE_SB_SendMsg	

Returns

See also

CFE_EVS_SendEvent, CFE_EVS_SendTimedEvent

Referenced by CFE_ES_RegisterCDS(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLast↩SenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB↩_UnsubscribeFull().

13.54.2.6 CFE_EVS_SendTimedEvent()

```
int32 int32 int32 CFE_EVS_SendTimedEvent (
            CFE_TIME_SysTime_t Time,
            uint16 EventID,
            uint16 EventType,
            const char * Spec,
             ... )
```

Description

This routine is the same as CFE_EVS_SendEvent except that the caller specifies the event time instead of having the EVS use the current spacecraft time. This routine should be used in situations where an error condition is detected at one time, but the event message is reported at a later time.

Assumptions, External Events, and Notes:

None

Parameters

in	*Time*	The time to include in the event. This will usually be a time returned by the function CFE_TIME_GetTime().
in	*EventID*	A numeric literal used to uniquely identify an application event. The `EventID` is defined and supplied by the application sending the event.
in	*EventType*	A numeric literal used to classify an event, one of: • CFE_EVS_EventType_DEBUG • CFE_EVS_EventType_INFORMATION • CFE_EVS_EventType_ERROR • CFE_EVS_EventType_CRITICAL
in	*Spec*	A pointer to a null terminated text string describing the output format for the event. This is the same type of format string used for the ANSI `printf` function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter CFE_MISSION_EVS_MAX_MESSAGE_LENGTH. Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	
Any of the error codes from CFE_SB_SendMsg	

Returns

See also

CFE_EVS_SendEvent, CFE_EVS_SendEventWithAppID

13.54.2.7 CFE_EVS_Unregister()

```
int32 CFE_EVS_Unregister (
            void  )
```

Description

This routine un-registers the calling application from receiving event services and removes and deletes the calling applications filters and counters from the internal event service filter and counter tables if registered. Applications must call this routine as part of their orderly shutdown process.

Assumptions, External Events, and Notes:

None

CFE_SUCCESS	Operation was performed successfully
CFE_EVS_APP_NOT_REGISTERED	Calling application never previously called CFE_EVS_Register()
CFE_EVS_APP_ILLEGAL_APP_ID	Application ID returned by CFE_ES_GetAppIDByName is greater than CFE_PLATFORM_ES_MAX_APPLICATIONS
Any of the error codes from CFE_ES_GetAppID	
Any of the error codes from CFE_ES_PutPoolBuf	

Returns

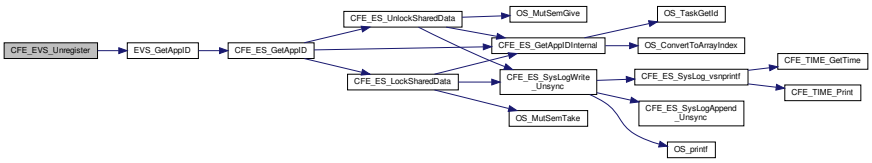
See also

CFE_EVS_Register

Definition at line 146 of file cfe_evs.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_GlobalData, CFE_EVS_UNDEF_APPID, CFE_SUCCESS, EVS_GetAppID(), and EVS_AppData_t::RegisterFlag.

Here is the call graph for this function:



13.55 cfe/fsw/cfe-core/src/inc/cfe_evs_events.h File Reference

Macros

- #define CFE_EVS_MAX_EID 43
- #define CFE_EVS_NOOP_EID 0 /∗ Noop event identifier ∗/

  'No-op command'
- #define CFE_EVS_STARTUP_EID 1

  'cFE EVS Initialized'
- #define CFE_EVS_ERR_WRLOGFILE_EID 2

  'Write Log File Command Error: OS_write = 0x%08X, filename = %s'
- #define CFE_EVS_ERR_CRLOGFILE_EID 3

  'Write Log File Command Error: OS_creat = 0x%08X, filename = %s'

- #define CFE_EVS_ERR_MSGID_EID 5

  'Invalid command packet, Message ID = 0x%08X'
- #define CFE_EVS_ERR_EVTIDNOREGS_EID 6

  '%s Event ID %d not registered for filtering: CC = %lu'
- #define CFE_EVS_ERR_APPNOREGS_EID 7

  '%s not registered with EVS: CC = %lu'
- #define CFE_EVS_ERR_ILLAPPIDRANGE_EID 8

  'Illegal application ID %d retrieved for %s: CC = %lu'
- #define CFE_EVS_ERR_NOAPPIDFOUND_EID 9

  'Unable to retrieve application ID for %s: CC = %lu'
- #define CFE_EVS_ERR_ILLEGALFMTMOD_EID 10

  'Set Event Format Mode Command: Invalid Event Format Mode = 0x%02x'
- #define CFE_EVS_ERR_MAXREGSFILTER_EID 11

  'Add Filter Command: number of registered filters has reached max = %d'
- #define CFE_EVS_ERR_WRDATFILE_EID 12

  'Write App Data Command Error: OS_write = 0x%08X, filename = %s'
- #define CFE_EVS_ERR_CRDATFILE_EID 13

  'Write App Data Command Error: OS_creat = 0x%08X, filename = %s'
- #define CFE_EVS_ERR_CC_EID 15

  'Invalid command code - ID = 0x%08x, CC = %d'
- #define CFE_EVS_RSTCNT_EID 16

  'Reset Counters Command Received'
- #define CFE_EVS_SETFILTERMSK_EID 17

  'Set Filter Mask Command Received with AppName=%s, EventID=0x%08x, Mask=0x%04x'
- #define CFE_EVS_ENAPORT_EID 18

  'Enable Ports Command Received with Port Bit Mask = 0x%02x'
- #define CFE_EVS_DISPORT_EID 19

  'Disable Ports Command Received with Port Bit Mask = 0x%02x'
- #define CFE_EVS_ENAEVTTYPE_EID 20

  'Enable Event Type Command Received with Event Type Bit Mask = 0x%02x'
- #define CFE_EVS_DISEVTTYPE_EID 21

  'Disable Event Type Command Received with Event Type Bit Mask = 0x%02x'
- #define CFE_EVS_SETEVTFMTMOD_EID 22

  'Set Event Format Mode Command Received with Mode = 0x%02x'
- #define CFE_EVS_ENAAPPEVTTYPE_EID 23

  'Enable App Event Type Command Received with AppName = %s, EventType Bit Mask = 0x%02x'
- #define CFE_EVS_DISAPPENTTYPE_EID 24

  'Disable App Event Type Command Received with AppName = %s, EventType Bit Mask = 0x%02x'
- #define CFE_EVS_ENAAPPEVT_EID 25

  'Enable App Events Command Received with AppName = %s'
- #define CFE_EVS_DISAPPEVT_EID 26

  'Disable App Events Command Received with AppName = %s'
- #define CFE_EVS_RSTEVTCNT_EID 27

  'Reset Event Counter Command Received with AppName = %s'
- #define CFE_EVS_RSTFILTER_EID 28

'Reset Filter Command Received with AppName = %s, EventID = 0x%08x'

- #define CFE_EVS_RSTALLFILTER_EID 29

  'Reset All Filters Command Received with AppName = %s'

- #define CFE_EVS_ADDFILTER_EID 30

  'Add Filter Command Received with AppName = %s, EventID = 0x%08x, Mask = 0x%04x'

- #define CFE_EVS_DELFILTER_EID 31

  'Delete Filter Command Received with AppName = %s, EventID = 0x%08x'

- #define CFE_EVS_WRDAT_EID 32

  'Write App Data Command: %d application data entries written to %s'

- #define CFE_EVS_WRLOG_EID 33

  'Write Log File Command: %d event log entries written to %s'

- #define CFE_EVS_NO_LOGSET_EID 34

  'Set Log Mode Command: Event Log is Disabled'

- #define CFE_EVS_NO_LOGCLR_EID 35

  'Clear Log Command: Event Log is Disabled'

- #define CFE_EVS_NO_LOGWR_EID 36

  'Write Log Command: Event Log is Disabled'

- #define CFE_EVS_EVT_FILTERED_EID 37

  'Add Filter Command:AppName = %s, EventID = 0x%08x is already registered for filtering'

- #define CFE_EVS_LOGMODE_EID 38

  'Set Log Mode Command Error: Log Mode = %d'

- #define CFE_EVS_ERR_LOGMODE_EID 39

  'Set Log Mode Command Error: Log Mode = %d'

- #define CFE_EVS_ERR_INVALID_BITMASK_EID 40

  'Bit Mask = 0x%X out of range: CC = %lu'

- #define CFE_EVS_ERR_UNREGISTERED_EVS_APP 41

  'App %s not registered with Event Services. Unable to send event'

- #define CFE_EVS_FILTER_MAX_EID 42

  'Max filter count reached, AppName = %s, EventID = 0x%08x: Filter locked until reset'

- #define CFE_EVS_LEN_ERR_EID 43

  'Invalid cmd length: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'

13.55.1 Macro Definition Documentation

13.55.1.1 CFE_EVS_ADDFILTER_EID

```
#define CFE_EVS_ADDFILTER_EID 30
```

Event Message `'Add Filter Command Received with AppName = %s, EventID = 0x%08x, Mask = 0x%04x'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Add Filter" command.

The `AppName` field identifies the Application who is getting the new filter, the `EventID` field identifies the Event Identifier, in hex, that is getting the filter, and the `Mask` field specifies, in hex, what the binary filter mask has been set to.

Definition at line 490 of file cfe_evs_events.h.

Referenced by CFE_EVS_AddEventFilterCmd().

13.55.1.2 CFE_EVS_DELFILTER_EID

```
#define CFE_EVS_DELFILTER_EID 31
```

Event Message `'Delete Filter Command Received with AppName = %s, EventID = 0x%08x'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Delete Filter" command.

The `AppName` field identifies the Application who is getting the filter removed, the `EventID` field identifies the Event Identifier, in hex, whose filter is being deleted.

Definition at line 504 of file cfe_evs_events.h.

Referenced by CFE_EVS_DeleteEventFilterCmd().

13.55.1.3 CFE_EVS_DISAPPENTTYPE_EID

```
#define CFE_EVS_DISAPPENTTYPE_EID 24
```

Event Message `'Disable App Event Type Command Received with AppName = %s, Event↩ Type Bit Mask = 0x%02x'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Disable Application Event Types" command.

The `AppName` field identifies the Application whose Event Type Disable status has changed and the `Mask` field specifies (in hex) the Event Types that have been disabled. Mask bits are defined by CFE_EVS_DEBUG_BIT, CFE_EVS_↩ INFORMATION_BIT, CFE_EVS_ERROR_BIT and CFE_EVS_CRITICAL_BIT.

Definition at line 409 of file cfe_evs_events.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd().

13.55.1.4 CFE_EVS_DISAPPEVT_EID

```
#define CFE_EVS_DISAPPEVT_EID 26
```

Event Message `'Disable App Events Command Received with AppName = %s'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Disable Application Events" command.

The `AppName` field identifies the Application whose Events have been Disabled.

Definition at line 435 of file cfe_evs_events.h.

Referenced by CFE_EVS_DisableAppEventsCmd().

13.55.1.5 CFE_EVS_DISEVTTYPE_EID

```
#define CFE_EVS_DISEVTTYPE_EID 21
```

Event Message `'Disable Event Type Command Received with Event Type Bit Mask = 0x%02x'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the "Disable Event Type" command.

The `Mask` field identifies the Event Types that are disabled. Mask bits are defined by CFE_EVS_DEBUG_BIT, CFE_EVS_INFORMATION_BIT, CFE_EVS_ERROR_BIT and CFE_EVS_CRITICAL_BIT.

Definition at line 363 of file cfe_evs_events.h.

Referenced by CFE_EVS_DisableEventTypeCmd().

13.55.1.6 CFE_EVS_DISPORT_EID

```
#define CFE_EVS_DISPORT_EID 19
```

Event Message `'Disable Ports Command Received with Port Bit Mask = 0x%02x'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the "Disable Ports" command.

The `Mask` field identifies (in hex) the ports are to be disabled. Mask bits are defined by CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT and CFE_EVS_PORT4_BIT.

Definition at line 333 of file cfe_evs_events.h.

Referenced by CFE_EVS_DisablePortsCmd().

13.55.1.7 CFE_EVS_ENAAPPEVT_EID

```
#define CFE_EVS_ENAAPPEVT_EID 25
```

Event Message `'Enable App Events Command Received with AppName = %s'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Enable Application Events" command.

The `AppName` field identifies the Application whose Events have been Enabled.

Definition at line 422 of file cfe_evs_events.h.

Referenced by CFE_EVS_EnableAppEventsCmd().

13.55.1.8 CFE_EVS_ENAAPPEVTTYPE_EID

```
#define CFE_EVS_ENAAPPEVTTYPE_EID 23
```

Event Message `'Enable App Event Type Command Received with AppName = %s, Event↩ Type Bit Mask = 0x%02x'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Enable Application Event Types" command.

The `AppName` field identifies the Application whose Event Type Enable status has changed and the `Mask` field specifies (in hex) the Event Types that have been enabled. Mask bits are defined by CFE_EVS_DEBUG_BIT, CFE_EVS_INF↩ ORMATION_BIT, CFE_EVS_ERROR_BIT and CFE_EVS_CRITICAL_BIT.

Definition at line 393 of file cfe_evs_events.h.

Referenced by CFE_EVS_EnableAppEventTypeCmd().

13.55.1.9 CFE_EVS_ENAEVTTYPE_EID

```
#define CFE_EVS_ENAEVTTYPE_EID 20
```

Event Message `'Enable Event Type Command Received with Event Type Bit Mask = 0x%02x'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the "Enable Event Type" command.

The `Mask` field identifies the Event Types that are enabled. Mask bits are defined by CFE_EVS_DEBUG_BIT, CFE_EVS_INFORMATION_BIT, CFE_EVS_ERROR_BIT and CFE_EVS_CRITICAL_BIT.

Definition at line 348 of file cfe_evs_events.h.

Referenced by CFE_EVS_EnableEventTypeCmd().

13.55.1.10 CFE_EVS_ENAPORT_EID

```
#define CFE_EVS_ENAPORT_EID 18
```

Event Message `'Enable Ports Command Received with Port Bit Mask = 0x%02x'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of the "Enable Ports" command.

The `Mask` field identifies the ports that are enabled. Mask bits are defined by CFE_EVS_PORT1_BIT, CFE_EVS_PORT2_BIT, CFE_EVS_PORT3_BIT and CFE_EVS_PORT4_BIT.

Definition at line 319 of file cfe_evs_events.h.

Referenced by CFE_EVS_EnablePortsCmd().

13.55.1.11 CFE_EVS_ERR_APPNOREGS_EID

```
#define CFE_EVS_ERR_APPNOREGS_EID 7
```

Event Message `'%s not registered with EVS: CC = %lu'`

Type: ERROR

Cause:

This event message is generated when the specified command identifies an Application that has not been registered with the cFE Event Services.

The `CC` field contains the Command Code whose processing resulted in the generation of the event message. Possible values are CFE_EVS_SET_FILTER_CC, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_RESET_APP_COUNTER_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_ADD_EVENT_FILTER_CC, or CFE_EVS_DELETE_EVENT_FILTER_CC.

Definition at line 157 of file cfe_evs_events.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

13.55.1.12 CFE_EVS_ERR_CC_EID

```
#define CFE_EVS_ERR_CC_EID 15
```

Event Message `'Invalid command code - ID = 0x%08x, CC = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_EVS_CMD_MID message ID has arrived but whose Command Code is not one of the specified accepted command codes specified. This problem is most likely to occur when:

1. A Message ID meant for another Application became corrupted and was set equal to CFE_EVS_CMD_MID.
2. The Command Code field in the Message became corrupted.
3. The command database at the ground station has been corrupted.

The `ID` field in the event message specifies the Message ID (in hex) and the `CC` field specifies the Command Code (in decimal) found in the message.

Definition at line 278 of file cfe_evs_events.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.55.1.13 CFE_EVS_ERR_CRDATFILE_EID

```
#define CFE_EVS_ERR_CRDATFILE_EID 13
```

Event Message `'Write App Data Command Error:  OS_creat = 0x%08X, filename = %s'`

Type: ERROR

Cause:

This event message is generated when a filesystem error occurred when attempting to create the file that is to hold the event registry data.

The message text identifies the registry filename and specifies the return value, in hex, from the system function call. The expected return value is a file handle, which in this case should be a relatively small positive number. Error codes are negative.

Definition at line 258 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

13.55.1.14 CFE_EVS_ERR_CRLOGFILE_EID

```
#define CFE_EVS_ERR_CRLOGFILE_EID 3
```

Event Message `'Write Log File Command Error:  OS_creat = 0x%08X, filename = %s'`

Type: ERROR

Cause:

This event message is generated when a filesystem error occurred when attempting to create the file that is to hold the event message log.

The message text identifies the event log filename and specifies the return value, in hex, from the system function call. The expected return value is a file handle, which in this case should be a relatively small positive number. Error codes are negative.

Definition at line 104 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

13.55.1.15 CFE_EVS_ERR_EVTIDNOREGS_EID

```
#define CFE_EVS_ERR_EVTIDNOREGS_EID 6
```

Event Message `'%s Event ID %d not registered for filtering:  CC = %lu'`

Type: ERROR

Cause:

This event message is generated when the specified command identifies an Application and Event ID combination that is not found in the Events Registry.

The `%s` string contains the command specified Application Name the `Event ID` field identifies the command specified EventID (in decimal) that was not found in the Events Registry. The `CC` field specifies the Command Code whose processing generated the event message. It can be equal to either CFE_EVS_SET_FILTER_CC, CFE_EVS_RESET_FILTER_CC, or CFE_EVS_DELETE_EVENT_FILTER_CC.

Definition at line 140 of file cfe_evs_events.h.

Referenced by CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

13.55.1.16 CFE_EVS_ERR_ILLAPPIDRANGE_EID

```
#define CFE_EVS_ERR_ILLAPPIDRANGE_EID 8
```

Event Message `'Illegal application ID %d retrieved for %s:  CC = %lu'`

Type: ERROR

Cause:

This event message is generated when the specified command identifies an Application whose name is found in the Events Registry but does not appear to be properly registered with the cFE Executive Services.

The `CC` field contains the Command Code whose processing resulted in the generation of the event message. Possible values are CFE_EVS_SET_FILTER_CC, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_RESET_APP_COUNTER_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_ADD_EVENT_FILTER_CC, or CFE_EVS_DELETE_EVENT_FILTER_CC.

Definition at line 174 of file cfe_evs_events.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

13.55.1.17 CFE_EVS_ERR_ILLEGALFMTMOD_EID

```
#define CFE_EVS_ERR_ILLEGALFMTMOD_EID 10
```

Event Message `'Set Event Format Mode Command:  Invalid Event Format Mode = 0x%02x'`

Type: ERROR

Cause:

This event message is generated when a "Set Event Format Mode" command message has arrived and the CFE_EVS_SetLogMode_Payload_t::LogMode field is equal to neither CFE_EVS_MsgFormat_SHORT or CFE_EVS_MsgFormat_LONG. These are the only allowed values for the mode field.

The `Mode` field in the event message identifies the Mode value (in hex) that was found in the message.

Definition at line 208 of file cfe_evs_events.h.

Referenced by CFE_EVS_SetEventFormatModeCmd().

13.55.1.18 CFE_EVS_ERR_INVALID_BITMASK_EID

```
#define CFE_EVS_ERR_INVALID_BITMASK_EID 40
```

Event Message `'Bit Mask = 0x%X out of range:  CC = %lu'`

Type: ERROR

Cause:

This event message is generated when the bit mask passed in is equal to zero or greater than 0x0F, because a bit mask of zero does nothing, and a bitmask of greater than 0x0F is invalid.

Definition at line 641 of file cfe_evs_events.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_DisableEventTypeCmd(), CFE_EVS_DisablePortsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_EnableEventTypeCmd(), and CFE_EVS_EnablePortsCmd().

13.55.1.19 CFE_EVS_ERR_LOGMODE_EID

```
#define CFE_EVS_ERR_LOGMODE_EID 39
```

Event Message `'Set Log Mode Command Error:  Log Mode = %d`

Type: ERROR

Cause:

This event message is generated when a "Set Log Mode" command is received that specifies an invalid Log Mode command argument.

The event text identifies the invalid Log Mode command argument. Valid Log Mode command arguments are: CFE_↩ EVS_LOG_OVERWRITE or CFE_EVS_LOG_DISCARD.

Definition at line 629 of file cfe_evs_events.h.

Referenced by CFE_EVS_SetLogModeCmd().

13.55.1.20 CFE_EVS_ERR_MAXREGSFILTER_EID

```
#define CFE_EVS_ERR_MAXREGSFILTER_EID 11
```

Event Message `'Add Filter Command:  number of registered filters has reached max = %d'`

Type: ERROR

Cause:

This event message is generated upon receipt of an "Add Filter" command and the specified Application has already reached the maximum number of filters allowed (CFE_PLATFORM_EVS_MAX_EVENT_FILTERS).

The `max` field in the event message identifies the maximum number of event filters allowed per Application. This value should be equal to the configuration parameter CFE_PLATFORM_EVS_MAX_EVENT_FILTERS.

Definition at line 225 of file cfe_evs_events.h.

Referenced by CFE_EVS_AddEventFilterCmd().

13.55.1.21 CFE_EVS_ERR_MSGID_EID

```
#define CFE_EVS_ERR_MSGID_EID 5
```

Event Message `'Invalid command packet, Message ID = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when a message has arrived on the cFE Event Services Application's Message Pipe that has a Message ID that is neither CFE_EVS_CMD_MID or CFE_EVS_SEND_HK_MID. Most likely, the cFE Software Bus routing table has become corrupt and is sending messages targeted for other Applications to the cFE Event Services Application.

The `ID` field in the event message identifies the message ID (in hex) that was found in the message.

Definition at line 123 of file cfe_evs_events.h.

Referenced by CFE_EVS_ProcessCommandPacket().

13.55.1.22 CFE_EVS_ERR_NOAPPIDFOUND_EID

```
#define CFE_EVS_ERR_NOAPPIDFOUND_EID 9
```

Event Message `'Unable to retrieve application ID for %s:  CC = %lu'`

Type: ERROR

Cause:

This event message is generated when the specified command contains an Application name that is apparently found in the Events Registry but does not appear to be registered with the cFE Executive Services.

The `CC` field contains the Command Code whose processing resulted in the generation of the event message. Possible values are CFE_EVS_SET_FILTER_CC, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC, CFE_EVS_RESET_APP_COUNTER_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_ADD_EVENT_FILTER_CC, or CFE_EVS_DELETE_EVENT_FILTER_CC.

Definition at line 191 of file cfe_evs_events.h.

Referenced by CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), and CFE_EVS_SetFilterCmd().

13.55.1.23 CFE_EVS_ERR_UNREGISTERED_EVS_APP

```
#define CFE_EVS_ERR_UNREGISTERED_EVS_APP 41
```

Event Message `'App %s not registered with Event Services.  Unable to send event'`

Type: ERROR

Cause:

This event message is generated when an event message has been requested to be sent by an Application that has not registered itself with cFE Event Services.

Definition at line 653 of file cfe_evs_events.h.

Referenced by EVS_NotRegistered().

13.55.1.24 CFE_EVS_ERR_WRDATFILE_EID

```
#define CFE_EVS_ERR_WRDATFILE_EID 12
```

Event Message `'Write App Data Command Error:  OS_write = 0x%08X, filename = %s'`

Type: ERROR

Cause:

This event message is generated when a filesystem error occurred while writing the contents of the event registry to a file.

The message text identifies the registry filename and specifies the return value, in hex, from the system function call. The expected return value is the number of bytes written, which in this case should be equal to the size of a CFE_EVS_AppDataFile_t structure. Error codes are negative.

Definition at line 242 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

13.55.1.25 CFE_EVS_ERR_WRLOGFILE_EID

```
#define CFE_EVS_ERR_WRLOGFILE_EID 2
```

Event Message `'Write Log File Command Error:  OS_write = 0x%08X, filename = %s'`

Type: ERROR

Cause:

This event message is generated when a filesystem error occurred while writing the contents of the event message log to a file.

The message text identifies the event log filename and specifies the return value, in hex, from the system function call. The expected return value is the number of bytes written, which in this case should be equal to the size of a CFE_EVS_LongEventTlm_t structure. Error codes are negative.

Definition at line 88 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

13.55.1.26 CFE_EVS_EVT_FILTERED_EID

```
#define CFE_EVS_EVT_FILTERED_EID 37
```

Event Message `'Add Filter Command:AppName = %s, EventID = 0x%08x is already registered for filtering'`

Type: ERROR

Cause:

This event message is generated when an "Add Filter" command was received specifying an Event ID that has already had a filter added.

The `AppName` field identifies the Application whose filter was to be added and the `EventID` field identifies, in hex, the Event ID that the command was trying to add a filter for.

Definition at line 600 of file cfe_evs_events.h.

Referenced by CFE_EVS_AddEventFilterCmd().

13.55.1.27 CFE_EVS_FILTER_MAX_EID

```
#define CFE_EVS_FILTER_MAX_EID 42
```

Event Message `'Max filter count reached, AppName = %s, EventID = 0x%08x: Filter locked until reset'`

Type: INFORMATIONAL

Cause:

This event message is generated when the filtering count for a specific App and Event ID reaches CFE_EVS_MAX_FILTER_COUNT The filtered event will no longer be received until the reset counter is reset via a "Reset an Event Filter for an Application" or a "Reset All Event Filters for an Application"

The `AppName` field identifies the Application and the `EventID` field identifies, in hex, the Event ID for the filter whose maxium was reached.

Definition at line 670 of file cfe_evs_events.h.

Referenced by EVS_IsFiltered().

13.55.1.28 CFE_EVS_LEN_ERR_EID

```
#define CFE_EVS_LEN_ERR_EID 43
```

Event Message `'Invalid cmd length: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_EVS_CMD_MID message ID has arrived but whose packet length does not match the expected length for the specified command code.

The `ID` field in the event message specifies the Message ID (in hex), the `CC` field specifies the Command Code (in decimal), the `Exp` Len field specified the Expected Length (in decimal), and `Len` specifies the message Length (in decimal) found in the message.

Definition at line 688 of file cfe_evs_events.h.

Referenced by CFE_EVS_VerifyCmdLength().

13.55.1.29 CFE_EVS_LOGMODE_EID

```
#define CFE_EVS_LOGMODE_EID 38
```

Event Message `'Set Log Mode Command Error:  Log Mode = %d'`

Type: DEBUG

Cause:

This event message is generated when a "Set Log Mode" command is completed successfully.

The event text identifies the Log Mode command argument. Valid Log Mode command arguments are: CFE_EVS_L↩OG_OVERWRITE or CFE_EVS_LOG_DISCARD.

Definition at line 614 of file cfe_evs_events.h.

Referenced by CFE_EVS_SetLogModeCmd().

13.55.1.30 CFE_EVS_MAX_EID

```
#define CFE_EVS_MAX_EID 43
```

Definition at line 45 of file cfe_evs_events.h.

13.55.1.31 CFE_EVS_NO_LOGCLR_EID

```
#define CFE_EVS_NO_LOGCLR_EID 35
```

Event Message `'Clear Log Command:  Event Log is Disabled'`

Type: ERROR

Cause:

This event message is generated upon receipt of a "Clear Log" command when the use of the Event Log has been disabled. To enable the Event Log, the cFE code must be compiled for the target with the **CFE_PLATFORM_EVS_↩LOG_ON** macro defined. The EVS task must also succeed during task initialization in acquiring a pointer to the cFE reset area and in the creation of a serializing semaphore to control access to the Event Log.

Definition at line 568 of file cfe_evs_events.h.

Referenced by CFE_EVS_ClearLogCmd().

13.55.1.32 CFE_EVS_NO_LOGSET_EID

```
#define CFE_EVS_NO_LOGSET_EID 34
```

Event Message 'Set Log Mode Command: Event Log is Disabled'

Type: ERROR

Cause:

This event message is generated upon receipt of a "Set Log Mode" command when the use of the Event Log has been disabled. To enable the Event Log, the cFE code must be compiled for the target with the **CFE_PLATFORM_EVS_↩ LOG_ON** macro defined. The EVS task must also succeed during task initialization in acquiring a pointer to the cFE reset area and in the creation of a serializing semaphore to control access to the Event Log.

Definition at line 551 of file cfe_evs_events.h.

Referenced by CFE_EVS_SetLogModeCmd().

13.55.1.33 CFE_EVS_NO_LOGWR_EID

```
#define CFE_EVS_NO_LOGWR_EID 36
```

Event Message 'Write Log Command: Event Log is Disabled'

Type: ERROR

Cause:

This event message is generated upon receipt of a "Write Log" command when the use of the Event Log has been disabled. To enable the Event Log, the cFE code must be compiled for the target with the **CFE_PLATFORM_EVS_↩ LOG_ON** macro defined. The EVS task must also succeed during task initialization in acquiring a pointer to the cFE reset area and in the creation of a serializing semaphore to control access to the Event Log.

Definition at line 585 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

13.55.1.34 CFE_EVS_NOOP_EID

```
#define CFE_EVS_NOOP_EID 0 /* Noop event identifier */
```

Event Message `'No-op command'`

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Event Services NO-OP command

Definition at line 59 of file cfe_evs_events.h.

Referenced by CFE_EVS_NoopCmd().

13.55.1.35 CFE_EVS_RSTALLFILTER_EID

```
#define CFE_EVS_RSTALLFILTER_EID 29
```

Event Message `'Reset All Filters Command Received with AppName = %s'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Reset Application Event Message Filters" command.

The `AppName` field identifies the Application whose entire set of Event Filters has been reset.

Definition at line 475 of file cfe_evs_events.h.

Referenced by CFE_EVS_ResetAllFiltersCmd().

13.55.1.36 CFE_EVS_RSTCNT_EID

```
#define CFE_EVS_RSTCNT_EID 16
```

Event Message `'Reset Counters Command Received'`

Type: DEBUG

Cause:

This event message is always automatically issued in response to a cFE Event Services Reset Counters command

Definition at line 290 of file cfe_evs_events.h.

Referenced by CFE_EVS_ResetCountersCmd().

13.55.1.37 CFE_EVS_RSTEVTCNT_EID

```
#define CFE_EVS_RSTEVTCNT_EID 27
```

Event Message `'Reset Event Counter Command Received with AppName = %s'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Reset Application Event Counter" command.

The `AppName` field identifies the Application whose Event Counter has been reset.

Definition at line 448 of file cfe_evs_events.h.

Referenced by CFE_EVS_ResetAppCounterCmd().

13.55.1.38 CFE_EVS_RSTFILTER_EID

```
#define CFE_EVS_RSTFILTER_EID 28
```

Event Message `'Reset Filter Command Received with AppName = %s, EventID = 0x%08x'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Reset Application Event Message Filter" command.

The `AppName` field identifies the Application whose Event Message Filter has been reset and the `EventID` field identifies the specific event message whose filter has been reset.

Definition at line 462 of file cfe_evs_events.h.

Referenced by CFE_EVS_ResetFilterCmd().

13.55.1.39 CFE_EVS_SETEVTFMTMOD_EID

```
#define CFE_EVS_SETEVTFMTMOD_EID 22
```

Event Message `'Set Event Format Mode Command Received with Mode = 0x%02x'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Set Event Format Mode" command.

The `Mode` field contains the newly chosen Event Format Mode (specified in hex). Acceptable values for this parameter are: CFE_EVS_MsgFormat_SHORT or CFE_EVS_MsgFormat_LONG

Definition at line 377 of file cfe_evs_events.h.

Referenced by CFE_EVS_SetEventFormatModeCmd().

13.55.1.40 CFE_EVS_SETFILTERMSK_EID

```
#define CFE_EVS_SETFILTERMSK_EID 17
```

Event Message `'Set Filter Mask Command Received with AppName=%s, EventID=0x%08x, Mask=0x%04x'`

Type: DEBUG

Cause:

This event message is issued upon successful processing of a Set Filter Mask command.

The `AppName` field identifies the Application whose Filter Mask has been changed. The `EventID` field identifies the Event whose Filter Mask has been changed. The `Mask` field identifies the new Mask value associated with the specified event.

Definition at line 305 of file cfe_evs_events.h.

Referenced by CFE_EVS_SetFilterCmd().

13.55.1.41 CFE_EVS_STARTUP_EID

```
#define CFE_EVS_STARTUP_EID 1
```

Event Message `'cFE EVS Initialized'`

Type: INFORMATION

Cause:

This event message is always automatically issued when the Event Services Task completes its Initialization.

Definition at line 71 of file cfe_evs_events.h.

Referenced by CFE_EVS_TaskInit().

13.55.1.42 CFE_EVS_WRDAT_EID

```
#define CFE_EVS_WRDAT_EID 32
```

Event Message `'Write App Data Command: %d application data entries written to %s'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Write Event Services Application Information to File" command.

The message text identifies the event log filename and specifies the number, in decimal, of events written to the file.

Definition at line 519 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteAppDataFileCmd().

13.55.1.43 CFE_EVS_WRLOG_EID

```
#define CFE_EVS_WRLOG_EID 33
```

Event Message `'Write Log File Command: %d event log entries written to %s'`

Type: DEBUG

Cause:

This event message is generated upon successful completion of the "Write Event Log to File" command.

The message text identifies the event log filename and specifies the number, in decimal, of events written to the file.

Definition at line 534 of file cfe_evs_events.h.

Referenced by CFE_EVS_WriteLogDataFileCmd().

13.56 cfe/fsw/cfe-core/src/inc/cfe_evs_extern_typedefs.h File Reference

```
#include "common_types.h"
```

Typedefs

- typedef uint8 CFE_EVS_MsgFormat_Enum_t
  Identifies format of log messages.
- typedef uint8 CFE_EVS_LogMode_Enum_t
  Identifies handling of log messages after storage is filled.
- typedef uint16 CFE_EVS_EventType_Enum_t
  Identifies type of event message.
- typedef uint8 CFE_EVS_EventFilter_Enum_t
  Identifies event filter schemes.
- typedef uint8 CFE_EVS_EventOutput_Enum_t
  Identifies event output port.

Enumerations

- enum CFE_EVS_MsgFormat { CFE_EVS_MsgFormat_SHORT = 0, CFE_EVS_MsgFormat_LONG = 1 }
  Label definitions associated with CFE_EVS_MsgFormat_Enum_t.
- enum CFE_EVS_LogMode { CFE_EVS_LogMode_OVERWRITE = 0, CFE_EVS_LogMode_DISCARD = 1 }
  Label definitions associated with CFE_EVS_LogMode_Enum_t.
- enum CFE_EVS_EventType { CFE_EVS_EventType_DEBUG = 1, CFE_EVS_EventType_INFORMATION = 2, CFE_EVS_EventType_ERROR = 3, CFE_EVS_EventType_CRITICAL = 4 }
  Label definitions associated with CFE_EVS_EventType_Enum_t.
- enum CFE_EVS_EventFilter { CFE_EVS_EventFilter_BINARY = 0 }
  Label definitions associated with CFE_EVS_EventFilter_Enum_t.
- enum CFE_EVS_EventOutput { CFE_EVS_EventOutput_PORT1 = 1, CFE_EVS_EventOutput_PORT2 = 2, CFE_EVS_EventOutput_PORT3 = 3, CFE_EVS_EventOutput_PORT4 = 4 }
  Label definitions associated with CFE_EVS_EventOutput_Enum_t.

13.56.1 Typedef Documentation

13.56.1.1 CFE_EVS_EventFilter_Enum_t

```
typedef uint8 CFE_EVS_EventFilter_Enum_t
```

See also

enum CFE_EVS_EventFilter

Definition at line 142 of file cfe_evs_extern_typedefs.h.

13.56.1.2 CFE_EVS_EventOutput_Enum_t

```
typedef uint8 CFE_EVS_EventOutput_Enum_t
```

See also

enum CFE_EVS_EventOutput

Definition at line 178 of file cfe_evs_extern_typedefs.h.

13.56.1.3 CFE_EVS_EventType_Enum_t

```
typedef uint16 CFE_EVS_EventType_Enum_t
```

See also

enum CFE_EVS_EventType

Definition at line 121 of file cfe_evs_extern_typedefs.h.

13.56.1.4 CFE_EVS_LogMode_Enum_t

```
typedef uint8 CFE_EVS_LogMode_Enum_t
```

See also

enum CFE_EVS_LogMode

Definition at line 85 of file cfe_evs_extern_typedefs.h.

13.56.1.5 CFE_EVS_MsgFormat_Enum_t

```
typedef uint8 CFE_EVS_MsgFormat_Enum_t
```

See also

enum CFE_EVS_MsgFormat

Definition at line 59 of file cfe_evs_extern_typedefs.h.

13.56.2 Enumeration Type Documentation

13.56.2.1 CFE_EVS_EventFilter

```
enum CFE_EVS_EventFilter
```

Enumerator

CFE_EVS_EventFilter_BINARY	Binary event filter.

Definition at line 127 of file cfe_evs_extern_typedefs.h.

13.56.2.2 CFE_EVS_EventOutput

`enum CFE_EVS_EventOutput`

Enumerator

CFE_EVS_EventOutput_PORT1	Output Port 1.
CFE_EVS_EventOutput_PORT2	Output Port 2.
CFE_EVS_EventOutput_PORT3	Output Port 3.
CFE_EVS_EventOutput_PORT4	Output Port 4.

Definition at line 148 of file cfe_evs_extern_typedefs.h.

13.56.2.3 CFE_EVS_EventType

`enum CFE_EVS_EventType`

Enumerator

CFE_EVS_EventType_DEBUG	Events that are intended only for debugging, not nominal operations.
CFE_EVS_EventType_INFORMATION	Events that identify a state change or action that is not an error.
CFE_EVS_EventType_ERROR	Events that identify an error but are not catastrophic (e.g. - bad command.
CFE_EVS_EventType_CRITICAL	Events that identify errors that are unrecoverable autonomously.

Definition at line 91 of file cfe_evs_extern_typedefs.h.

13.56.2.4 CFE_EVS_LogMode

`enum CFE_EVS_LogMode`

Enumerator

CFE_EVS_LogMode_OVERWRITE	Overwrite Log Mode.
CFE_EVS_LogMode_DISCARD	Discard Log Mode.

Definition at line 65 of file cfe_evs_extern_typedefs.h.

13.56.2.5 CFE_EVS_MsgFormat

```
enum CFE_EVS_MsgFormat
```

Enumerator

CFE_EVS_MsgFormat_SHORT	Short Format Messages.
CFE_EVS_MsgFormat_LONG	Long Format Messages.

Definition at line 39 of file cfe_evs_extern_typedefs.h.

13.57 cfe/fsw/cfe-core/src/inc/cfe_evs_msg.h File Reference

```
#include "common_types.h"
#include "cfe_time.h"
#include "cfe_sb.h"
#include "cfe_es.h"
```

Data Structures

- struct CFE_EVS_NoArgsCmd_t

  Command with no additional arguments.
- struct CFE_EVS_LogFileCmd_Payload_t

  Write Event Log to File Command.
- struct CFE_EVS_WriteLogDataFile_t
- struct CFE_EVS_AppDataCmd_Payload_t

  Write Event Services Application Information to File Command.
- struct CFE_EVS_WriteAppDataFile_t
- struct CFE_EVS_SetLogMode_Payload_t

  Set Event Format Mode or Set Log Mode Commands.
- struct CFE_EVS_SetLogMode_t
- struct CFE_EVS_SetEventFormatMode_Payload_t

  Set Event Format Mode or Set Log Mode Commands.
- struct CFE_EVS_SetEventFormatMode_t
- struct CFE_EVS_BitMaskCmd_Payload_t

  Enable/Disable Events or Ports Commands.
- struct CFE_EVS_BitMaskCmd_t
- struct CFE_EVS_AppNameCmd_Payload_t

  Enable/Disable Application Events or Reset One or All Filter Counters.
- struct CFE_EVS_AppNameCmd_t
- struct CFE_EVS_AppNameEventIDCmd_Payload_t

Reset an Event Filter for an Application.

- struct CFE_EVS_AppNameEventIDCmd_t
- struct CFE_EVS_AppNameBitMaskCmd_Payload_t

Enable/Disable an Event Type for an Application.

- struct CFE_EVS_AppNameBitMaskCmd_t
- struct CFE_EVS_AppNameEventIDMaskCmd_Payload_t

Set, Add or Delete an Event Filter for an Application.

- struct CFE_EVS_AppNameEventIDMaskCmd_t
- struct CFE_EVS_AppTlmData_t
- struct CFE_EVS_HousekeepingTlm_Payload_t
- struct CFE_EVS_HousekeepingTlm_t
- struct CFE_EVS_PacketID_t
- struct CFE_EVS_LongEventTlm_Payload_t
- struct CFE_EVS_ShortEventTlm_Payload_t
- struct CFE_EVS_LongEventTlm_t
- struct CFE_EVS_ShortEventTlm_t

Macros

- #define CFE_EVS_DEBUG_BIT 0x0001
- #define CFE_EVS_INFORMATION_BIT 0x0002
- #define CFE_EVS_ERROR_BIT 0x0004
- #define CFE_EVS_CRITICAL_BIT 0x0008
- #define CFE_EVS_PORT1_BIT 0x0001
- #define CFE_EVS_PORT2_BIT 0x0002
- #define CFE_EVS_PORT3_BIT 0x0004
- #define CFE_EVS_PORT4_BIT 0x0008
- #define CFE_EVS_LOG_OVERWRITE 0
- #define CFE_EVS_LOG_DISCARD 1
- #define CFE_EVS_HK_TLM_LNGTH sizeof(CFE_EVS_TlmPkt_t)

Event Services Command Codes

- #define CFE_EVS_NOOP_CC 0
- #define CFE_EVS_RESET_COUNTERS_CC 1
- #define CFE_EVS_ENABLE_EVENT_TYPE_CC 2
- #define CFE_EVS_DISABLE_EVENT_TYPE_CC 3
- #define CFE_EVS_SET_EVENT_FORMAT_MODE_CC 4
- #define CFE_EVS_ENABLE_APP_EVENT_TYPE_CC 5
- #define CFE_EVS_DISABLE_APP_EVENT_TYPE_CC 6
- #define CFE_EVS_ENABLE_APP_EVENTS_CC 7
- #define CFE_EVS_DISABLE_APP_EVENTS_CC 8
- #define CFE_EVS_RESET_APP_COUNTER_CC 9
- #define CFE_EVS_SET_FILTER_CC 10
- #define CFE_EVS_ENABLE_PORTS_CC 11
- #define CFE_EVS_DISABLE_PORTS_CC 12
- #define CFE_EVS_RESET_FILTER_CC 13
- #define CFE_EVS_RESET_ALL_FILTERS_CC 14
- #define CFE_EVS_ADD_EVENT_FILTER_CC 15
- #define CFE_EVS_DELETE_EVENT_FILTER_CC 16
- #define CFE_EVS_WRITE_APP_DATA_FILE_CC 17
- #define CFE_EVS_WRITE_LOG_DATA_FILE_CC 18
- #define CFE_EVS_SET_LOG_MODE_CC 19
- #define CFE_EVS_CLEAR_LOG_CC 20

Typedefs

- typedef CFE_EVS_NoArgsCmd_t CFE_EVS_Noop_t
- typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ResetCounters_t
- typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ClearLog_t
- typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnablePorts_t
- typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisablePorts_t
- typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnableEventType_t
- typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisableEventType_t
- typedef CFE_EVS_AppNameCmd_t CFE_EVS_EnableAppEvents_t
- typedef CFE_EVS_AppNameCmd_t CFE_EVS_DisableAppEvents_t
- typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAppCounter_t
- typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAllFilters_t
- typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_ResetFilter_t
- typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_DeleteEventFilter_t
- typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_EnableAppEventType_t
- typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_DisableAppEventType_t
- typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_AddEventFilter_t
- typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_SetFilter_t
- typedef CFE_EVS_LongEventTlm_t CFE_EVS_Packet_t
- typedef CFE_EVS_HousekeepingTlm_t CFE_EVS_TlmPkt_t

13.57.1 Macro Definition Documentation

13.57.1.1 CFE_EVS_ADD_EVENT_FILTER_CC

```
#define CFE_EVS_ADD_EVENT_FILTER_CC 15
```

Name Add Application Event Filter

Description

This command adds the given filter for the given application identifier and event identifier. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **`$sc_$cpu_EVS_AddEvtFltr`**

Command Structure

CFE_EVS_AppNameEventIDMaskCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The generation of CFE_EVS_ADDFILTER_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

None.

See also

CFE_EVS_SET_FILTER_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE↩_EVS_DELETE_EVENT_FILTER_CC

Definition at line 719 of file cfe_evs_msg.h.

Referenced by CFE_EVS_AddEventFilterCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.2 CFE_EVS_CLEAR_LOG_CC

```
#define CFE_EVS_CLEAR_LOG_CC 20
```

Name Clear Event Log

Description

This command clears the contents of the local event log.

Command Mnemonic(s) **$sc_$cpu_EVS_ClrLog**

Command Structure

CFE_TBL_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length

  Evidence of failure may be found in the following telemetry:

  - **`$sc_$cpu_EVS_CMDEC` -** command error counter will increment
  - An Error specific event message

Criticality

Clearing the local event log is not particularly hazardous, as the result may be making available space to record valuable event data. However, inappropriately clearing the local event log could result in a loss of critical information. Note: the event log is a back-up log to the on-board recorder.

See also

CFE_EVS_WRITE_LOG_DATA_FILE_CC, CFE_EVS_SET_LOG_MODE_CC

Definition at line 896 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.1.3 CFE_EVS_CRITICAL_BIT

```
#define CFE_EVS_CRITICAL_BIT 0x0008
```

Definition at line 903 of file cfe_evs_msg.h.

Referenced by EVS_DisableTypes(), EVS_EnableTypes(), and EVS_IsFiltered().

13.57.1.4 CFE_EVS_DEBUG_BIT

```
#define CFE_EVS_DEBUG_BIT 0x0001
```

Definition at line 900 of file cfe_evs_msg.h.

Referenced by EVS_DisableTypes(), EVS_EnableTypes(), and EVS_IsFiltered().

13.57.1.5 CFE_EVS_DELETE_EVENT_FILTER_CC

```
#define CFE_EVS_DELETE_EVENT_FILTER_CC 16
```

Name Delete Application Event Filter

Description

This command removes the given filter for the given application identifier and event identifier. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **`$sc_$cpu_EVS_DelEvtFltr`**

Command Structure

CFE_EVS_AppNameEventIDCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The generation of CFE_EVS_DELFILTER_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range
  Evidence of failure may be found in the following telemetry:
  - **`$sc_$cpu_EVS_CMDEC`** - command error counter will increment
  - An Error specific event message

Criticality

None.

See also

CFE_EVS_SET_FILTER_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE↩_EVS_ADD_EVENT_FILTER_CC

Definition at line 755 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DeleteEventFilterCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.6 CFE_EVS_DISABLE_APP_EVENT_TYPE_CC

```
#define CFE_EVS_DISABLE_APP_EVENT_TYPE_CC 6
```

Name Disable Application Event Type

Description

This command disables the command specified event type for the command specified application, preventing the application from sending event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **`$sc_$cpu_EVS_DisAppEvtType`**, **`$sc_$cpu_EVS_DisAppEvtTypeMask`**

Command Structure

CFE_EVS_AppNameBitMaskCmd_t The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be disabled (or filtered) for the specified application. A zero in a bit position means the filtering state is unchanged for the specified application.

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The generation of CFE_EVS_DISAPPENTTYPE_EID debug event message
- The clearing of the Event Type Active Flag in The Event Type Active Flag in EVS App Data File

Error Conditions

This command may fail for the following reason(s):

- Invalid Event Type Selection
  Evidence of failure may be found in the following telemetry:
  - **`$sc_$cpu_EVS_CMDEC`** - command error counter will increment
  - An Error specific event message

Criticality

Disabling an application's event type is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately disabling an application's event type could result in a loss of critical information and missed behavior for the ground system.

See also

CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_ENABLE_AP↩
P_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC

Definition at line 369 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventTypeCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.7 CFE_EVS_DISABLE_APP_EVENTS_CC

```
#define CFE_EVS_DISABLE_APP_EVENTS_CC 8
```

Name Disable Event Services for an Application

Description

This command disables the command specified application from sending events through Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **`$sc_$cpu_EVS_DisAppEvGen`**

Command Structure

CFE_EVS_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The generation of CFE_EVS_DISAPPEVT_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range
  Evidence of failure may be found in the following telemetry:
  - **`$sc_$cpu_EVS_CMDEC`** - command error counter will increment
  - An Error specific event message

Criticality

Disabling an application's events is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately disabling an application's events could result in a loss of critical information and missed behavior for the ground system.

See also

CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_ENABLE_AP↩P_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC

Definition at line 451 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableAppEventsCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.8 CFE_EVS_DISABLE_EVENT_TYPE_CC

```
#define CFE_EVS_DISABLE_EVENT_TYPE_CC 3
```

Name Disable Event Type

Description

This command disables the command specified Event Type preventing event messages of this type to be sent through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, error and critical. This command is a global disable of a particular event type, it applies to all applications.

Command Mnemonic(s) **`$sc_$cpu_EVS_DisEventType`**, **`$sc_$cpu_EVS_DisEventTypeMask`**

Command Structure

CFE_EVS_BitMaskCmd_t The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be disabled (or filtered). A zero in a bit position means the filtering state is unchanged.

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC` -** command execution counter will increment
- The generation of CFE_EVS_DISEVTTYPE_EID debug message

Error Conditions

This command may fail for the following reason(s):

- Invalid Event Type selection

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_EVS_CMDEC` -** command error counter will increment
- An Error specific event message

Criticality

Disabling an event type is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately disabling an event type could result in a loss of critical information and missed behavior for the ground system.

See also

CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC

Definition at line 215 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisableEventTypeCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.9 CFE_EVS_DISABLE_PORTS_CC

```
#define CFE_EVS_DISABLE_PORTS_CC 12
```

Name Disable Event Services Output Ports

Description

This command disables the specified port from outputting event messages.

Command Mnemonic(s) **`$sc_$cpu_EVS_DisPort`**, **`$sc_$cpu_EVS_DisPortMask`**

Command Structure

CFE_EVS_BitMaskCmd_t The following bit positions apply to structure member named 'BitMask'. Bit 0 - Port 1 Bit 1 - Port 2 Bit 2 - Port 3 Bit 3 - Port 4 A one in a bit position means the port will be disabled. A zero in a bit position means the port state is unchanged.

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The generation of CFE_EVS_DISPORT_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Invalid PORT selection

  Evidence of failure may be found in the following telemetry:
  - **`$sc_$cpu_EVS_CMDEC`** - command error counter will increment
  - An Error specific event message

Criticality

None.

See also

CFE_EVS_ENABLE_PORTS_CC

Definition at line 611 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisablePortsCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.10 CFE_EVS_ENABLE_APP_EVENT_TYPE_CC

```
#define CFE_EVS_ENABLE_APP_EVENT_TYPE_CC 5
```

Name Enable Application Event Type

Description

This command enables the command specified event type for the command specified application, allowing the application to send event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **`$sc_$cpu_EVS_EnaAppEvtType`**, **`$sc_$cpu_EVS_EnaAppEvtTypeMask`**

Command Structure

CFE_EVS_AppNameBitMaskCmd_t The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be enabled (or unfiltered) for the specified application. A zero in a bit position means the filtering state is unchanged for the specified application.

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC` -** command execution counter will increment
- The generation of CFE_EVS_ENAAPPEVTTYPE_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid Event Type Selection
  Evidence of failure may be found in the following telemetry:
  - **`$sc_$cpu_EVS_CMDEC` -** command error counter will increment
  - An Error specific event message

Criticality

Enabling an application event type is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an application's event type could result in flooding of the ground system.

See also

CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_DISABLE_AP↩
P_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC

Definition at line 317 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EnableAppEventTypeCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.11 CFE_EVS_ENABLE_APP_EVENTS_CC

```
#define CFE_EVS_ENABLE_APP_EVENTS_CC 7
```

Name Enable Event Services for an Application

Description

This command enables the command specified application to send events through the Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **$sc_$cpu_EVS_EnaAppEvGen**

Command Structure

CFE_EVS_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_ENAAPPEVT_EID debug event message
- The setting of the Active Flag in The Active Flag in EVS App Data File

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

Enabling an application events is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an application's events could result in flooding of the ground system.

See also

CFE_EVS_ENABLE_EVENT_TYPE_CC, CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_ENABLE_AP↩P_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABLE_APP_EVENTS_↩CC

Definition at line 410 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EnableAppEventsCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.12 CFE_EVS_ENABLE_EVENT_TYPE_CC

```
#define CFE_EVS_ENABLE_EVENT_TYPE_CC 2
```

Name Enable Event Type

Description

This command enables the command specified Event Type allowing event messages of this type to be sent through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, error and critical. This command is a global enable of a particular event type, it applies to all applications.

Command Mnemonic(s) **`$sc_$cpu_EVS_EnaEventType`**, **`$sc_$cpu_EVS_EnaEventTypeMask`**

Command Structure

CFE_EVS_BitMaskCmd_t The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be enabled (or unfiltered). A zero in a bit position means the filtering state is unchanged.

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC` -** command execution counter will increment
- The generation of CFE_EVS_ENAEVTTYPE_EID debug message

Error Conditions

This command may fail for the following reason(s):

Invalid Event Type selection

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_EVS_CMDEC` -** command error counter will increment
- An Error specific event message

Criticality

Enabling an event type is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an event type could result in flooding of the system.

See also

CFE_EVS_DISABLE_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENT_TYPE_CC, CFE_EVS_DISABL↩E_APP_EVENT_TYPE_CC, CFE_EVS_ENABLE_APP_EVENTS_CC, CFE_EVS_DISABLE_APP_EVENTS_CC

Definition at line 165 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EnableEventTypeCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.13 CFE_EVS_ENABLE_PORTS_CC

```
#define CFE_EVS_ENABLE_PORTS_CC 11
```

Name Enable Event Services Output Ports

Description

This command enables the command specified port to output event messages

Command Mnemonic(s) **`$sc_$cpu_EVS_EnaPort`**, **`$sc_$cpu_EVS_EnaPortMask`**

Command Structure

CFE_EVS_BitMaskCmd_t The following bit positions apply to structure member named 'BitMask'. Bit 0 - Port 1 Bit 1 - Port 2 Bit 2 - Port 3 Bit 3 - Port 4 A one in a bit position means the port will be enabled. A zero in a bit position means the port state is unchanged.

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The generation of CFE_EVS_ENAPORT_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Invalid PORT selection

  Evidence of failure may be found in the following telemetry:
  - **`$sc_$cpu_EVS_CMDEC`** - command error counter will increment
  - An Error specific event message

Criticality

None.

See also

CFE_EVS_DISABLE_PORTS_CC

Definition at line 571 of file cfe_evs_msg.h.

Referenced by CFE_EVS_EnablePortsCmd(), and CFE_EVS_ProcessGroundCommand().

13.57.1.14 CFE_EVS_ERROR_BIT

```
#define CFE_EVS_ERROR_BIT 0x0004
```

Definition at line 902 of file cfe_evs_msg.h.

Referenced by EVS_DisableTypes(), EVS_EnableTypes(), and EVS_IsFiltered().

13.57.1.15 CFE_EVS_HK_TLM_LNGTH

```
#define CFE_EVS_HK_TLM_LNGTH sizeof(CFE_EVS_TlmPkt_t)
```

Definition at line 1250 of file cfe_evs_msg.h.

13.57.1.16 CFE_EVS_INFORMATION_BIT

```
#define CFE_EVS_INFORMATION_BIT 0x0002
```

Definition at line 901 of file cfe_evs_msg.h.

Referenced by EVS_DisableTypes(), EVS_EnableTypes(), and EVS_IsFiltered().

13.57.1.17 CFE_EVS_LOG_DISCARD

```
#define CFE_EVS_LOG_DISCARD 1
```

Definition at line 913 of file cfe_evs_msg.h.

13.57.1.18 CFE_EVS_LOG_OVERWRITE

```
#define CFE_EVS_LOG_OVERWRITE 0
```

Definition at line 912 of file cfe_evs_msg.h.

13.57.1.19 CFE_EVS_NOOP_CC

```
#define CFE_EVS_NOOP_CC 0
```

Name Event Services No-Op

Description

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Event Services task.

Command Mnemonic(s) **`$sc_$cpu_EVS_NOOP`**

Command Structure

CFE_TBL_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_EVS_CMDPC`** - command execution counter will increment
- The CFE_EVS_NOOP_EID informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Event Services receives the command, the event is sent (although it may be filtered by EVS itself) and the counter is incremented unconditionally.

Criticality

None

See also

Definition at line 79 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.1.20 CFE_EVS_PORT1_BIT

```
#define CFE_EVS_PORT1_BIT 0x0001
```

Definition at line 906 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisablePortsCmd(), CFE_EVS_EnablePortsCmd(), and EVS_SendViaPorts().

13.57.1.21 CFE_EVS_PORT2_BIT

```
#define CFE_EVS_PORT2_BIT 0x0002
```

Definition at line 907 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisablePortsCmd(), CFE_EVS_EnablePortsCmd(), and EVS_SendViaPorts().

13.57.1.22 CFE_EVS_PORT3_BIT

```
#define CFE_EVS_PORT3_BIT 0x0004
```

Definition at line 908 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisablePortsCmd(), CFE_EVS_EnablePortsCmd(), and EVS_SendViaPorts().

13.57.1.23 CFE_EVS_PORT4_BIT

```
#define CFE_EVS_PORT4_BIT 0x0008
```

Definition at line 909 of file cfe_evs_msg.h.

Referenced by CFE_EVS_DisablePortsCmd(), CFE_EVS_EnablePortsCmd(), and EVS_SendViaPorts().

13.57.1.24 CFE_EVS_RESET_ALL_FILTERS_CC

```
#define CFE_EVS_RESET_ALL_FILTERS_CC 14
```

Name Reset All Event Filters for an Application

Description

This command resets all of the command specified applications event filters. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **`$sc_$cpu_EVS_RstAllFltrs`**

Command Structure

CFE_EVS_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_RSTALLFILTER_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

None.

See also

CFE_EVS_SET_FILTER_CC, CFE_EVS_RESET_FILTER_CC, CFE_EVS_ADD_EVENT_FILTER_CC, CFE_↩ EVS_DELETE_EVENT_FILTER_CC

Definition at line 683 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand(), and CFE_EVS_ResetAllFiltersCmd().

13.57.1.25 CFE_EVS_RESET_APP_COUNTER_CC

```
#define CFE_EVS_RESET_APP_COUNTER_CC 9
```

Name Reset Application Event Counters

Description

This command sets the command specified application's event counter to zero. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **$sc_$cpu_EVS_RstAppCtrs**

Command Structure

CFE_EVS_AppNameCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_RSTEVTCNT_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range

  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter value that is reset by this command.

See also

CFE_EVS_RESET_COUNTERS_CC

Definition at line 489 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand(), and CFE_EVS_ResetAppCounterCmd().

13.57.1.26 CFE_EVS_RESET_COUNTERS_CC

```
#define CFE_EVS_RESET_COUNTERS_CC 1
```

Name Event Services Reset Counters

Description

This command resets the following counters within the Event Services housekeeping telemetry:

- Command Execution Counter ($sc_$cpu_EVS_CMDPC)
- Command Error Counter ($sc_$cpu_EVS_CMDEC)

Command Mnemonic(s) **$sc_$cpu_EVS_ResetCtrs**

Command Structure

CFE_TBL_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The CFE_EVS_RSTCNT_EID debug event message will be generated

Error Conditions

There are no error conditions for this command. If the Event Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

See also

CFE_EVS_RESET_APP_COUNTER_CC

Definition at line 116 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.1.27 CFE_EVS_RESET_FILTER_CC

```
#define CFE_EVS_RESET_FILTER_CC 13
```

Name Reset an Event Filter for an Application

Description

This command resets the command specified application's event filter for the command specified event ID. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **$sc_$cpu_EVS_RstBinFltrCtr**

Command Structure

CFE_EVS_AppNameEventIDCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_RSTFILTER_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range

  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

None.

See also

CFE_EVS_SET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_ADD_EVENT_FILTER_CC, CFE_EVS_DELETE_EVENT_FILTER_CC

Definition at line 647 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand(), and CFE_EVS_ResetFilterCmd().

13.57.1.28 CFE_EVS_SET_EVENT_FORMAT_MODE_CC

```
#define CFE_EVS_SET_EVENT_FORMAT_MODE_CC 4
```

Name Set Event Format Mode

Description

This command sets the event format mode to the command specified value. The event format mode may be either short or long. A short event format detaches the Event Data from the event message and only includes the following information in the event packet: Processor ID, Application ID, Event ID, and Event Type. Refer to section 5.3.3.4 for a description of the Event Service event packet contents. Event Data is defined to be data describing an Event that is supplied to the cFE Event Service. ASCII text strings are used as the primary format for Event Data because heritage ground systems use string compares as the basis for their automated alert systems. Two systems, ANSR and SERS were looked at for interface definitions. The short event format is used to accommodate experiences with limited telemetry bandwidth. The long event format includes all event information included within the short format along with the Event Data.

Command Mnemonic(s) **$sc_$cpu_EVS_SetEvtFmt**

Command Structure

CFE_EVS_SetLogMode_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_SETEVTFMTMOD_EID debug message

Error Conditions

This command may fail for the following reason(s): Invalid SB message (command) length Invalid MODE selection

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_EVS_CMDEC -** command error counter will increment
- An Error specific event message

Criticality

Setting the event format mode is not particularly hazardous, as the result may be saving necessary bandwidth. However, inappropriately setting the event format mode could result in a loss of information and missed behavior for the ground system

See also

Definition at line 264 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.1.29 CFE_EVS_SET_FILTER_CC

```
#define CFE_EVS_SET_FILTER_CC 10
```

Name Set Application Event Filter

Description

This command sets the command specified application's event filter mask to the command specified value for the command specified event. Note: In order for this command to take effect, applications must be registered for Event Service.

Command Mnemonic(s) **$sc_$cpu_EVS_SetBinFltrMask**

Command Structure

CFE_EVS_AppNameEventIDMaskCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_SETFILTERMSK_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Application selected is not registered to receive Event Service
- Application ID is out of range
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

Setting an application event filter mask is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately setting an application's event filter mask could result in a loss of critical information and missed behavior for the ground system or flooding of the ground system.

See also

CFE_EVS_RESET_FILTER_CC, CFE_EVS_RESET_ALL_FILTERS_CC, CFE_EVS_ADD_EVENT_FILTER_↩ CC, CFE_EVS_DELETE_EVENT_FILTER_CC

Definition at line 531 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand(), and CFE_EVS_SetFilterCmd().

13.57.1.30 CFE_EVS_SET_LOG_MODE_CC

```
#define CFE_EVS_SET_LOG_MODE_CC 19
```

Name Set Logging Mode

Description

This command sets the logging mode to the command specified value.

Command Mnemonic(s) **$sc_$cpu_EVS_SetLogMode**

Command Structure

CFE_EVS_SetLogMode_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_LOGMODE_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
- Invalid MODE selected
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

Setting the event logging mode is not particularly hazardous, as the result may be saving valuable event data. However, inappropriately setting the log mode could result in a loss of critical information. Note: the event log is a back-up log to the on-board recorder.

See also

CFE_EVS_WRITE_LOG_DATA_FILE_CC, CFE_EVS_CLEAR_LOG_CC

Definition at line 861 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.1.31 CFE_EVS_WRITE_APP_DATA_FILE_CC

```
#define CFE_EVS_WRITE_APP_DATA_FILE_CC 17
```

Name Write Event Services Application Information to File

Description

This command writes all application data to a file for all applications that have registered with the EVS. The application data includes the Application ID, Active Flag, Event Count, Event Types Active Flag, and Filter Data.

Command Mnemonic(s) **$sc_$cpu_EVS_WriteAppData2File**

Command Structure

CFE_EVS_WriteAppDataFile_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_WRDAT_EID debug event message
- The generation of the file written to

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

Writing a file is not particularly hazardous, but if proper file management is not taken, then the file system can fill up if this command is used repeatedly.

See also

CFE_EVS_WRITE_LOG_DATA_FILE_CC, CFE_EVS_SET_LOG_MODE_CC

Definition at line 791 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.1.32 CFE_EVS_WRITE_LOG_DATA_FILE_CC

```
#define CFE_EVS_WRITE_LOG_DATA_FILE_CC 18
```

Name Write Event Log to File

Description

This command requests the Event Service to generate a file containing the contents of the local event log.

Command Mnemonic(s) **$sc_$cpu_EVS_WriteLog2File**

Command Structure

CFE_EVS_WriteLogDataFile_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_EVS_CMDPC -** command execution counter will increment
- The generation of CFE_EVS_WRLOG_EID debug event message

Error Conditions

This command may fail for the following reason(s):

- Invalid SB message (command) length
  Evidence of failure may be found in the following telemetry:
  - **$sc_$cpu_EVS_CMDEC -** command error counter will increment
  - An Error specific event message

Criticality

Writing a file is not particularly hazardous, but if proper file management is not taken, then the file system can fill up if this command is used repeatedly.

See also

CFE_EVS_WRITE_APP_DATA_FILE_CC, CFE_EVS_SET_LOG_MODE_CC, CFE_EVS_CLEAR_LOG_CC

Definition at line 825 of file cfe_evs_msg.h.

Referenced by CFE_EVS_ProcessGroundCommand().

13.57.2 Typedef Documentation

13.57.2.1 CFE_EVS_AddEventFilter_t

```
typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_AddEventFilter_t
```

Definition at line 1121 of file cfe_evs_msg.h.

13.57.2.2 CFE_EVS_ClearLog_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ClearLog_t
```

Definition at line 931 of file cfe_evs_msg.h.

13.57.2.3 CFE_EVS_DeleteEventFilter_t

```
typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_DeleteEventFilter_t
```

Definition at line 1071 of file cfe_evs_msg.h.

13.57.2.4 CFE_EVS_DisableAppEvents_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_DisableAppEvents_t
```

Definition at line 1045 of file cfe_evs_msg.h.

13.57.2.5 CFE_EVS_DisableAppEventType_t

```
typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_DisableAppEventType_t
```

Definition at line 1096 of file cfe_evs_msg.h.

13.57.2.6 CFE_EVS_DisableEventType_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisableEventType_t
```

Definition at line 1021 of file cfe_evs_msg.h.

13.57.2.7 CFE_EVS_DisablePorts_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisablePorts_t
```

Definition at line 1019 of file cfe_evs_msg.h.

13.57.2.8 CFE_EVS_EnableAppEvents_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_EnableAppEvents_t
```

Definition at line 1044 of file cfe_evs_msg.h.

13.57.2.9 CFE_EVS_EnableAppEventType_t

```
typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_EnableAppEventType_t
```

Definition at line 1095 of file cfe_evs_msg.h.

13.57.2.10 CFE_EVS_EnableEventType_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnableEventType_t
```

Definition at line 1020 of file cfe_evs_msg.h.

13.57.2.11 CFE_EVS_EnablePorts_t

```
typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnablePorts_t
```

Definition at line 1018 of file cfe_evs_msg.h.

13.57.2.12 CFE_EVS_Noop_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_Noop_t
```

Definition at line 929 of file cfe_evs_msg.h.

13.57.2.13 CFE_EVS_Packet_t

```
typedef CFE_EVS_LongEventTlm_t CFE_EVS_Packet_t
```

Definition at line 1246 of file cfe_evs_msg.h.

13.57.2.14 CFE_EVS_ResetAllFilters_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAllFilters_t
```

Definition at line 1047 of file cfe_evs_msg.h.

13.57.2.15 CFE_EVS_ResetAppCounter_t

```
typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAppCounter_t
```

Definition at line 1046 of file cfe_evs_msg.h.

13.57.2.16 CFE_EVS_ResetCounters_t

```
typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ResetCounters_t
```

Definition at line 930 of file cfe_evs_msg.h.

13.57.2.17 CFE_EVS_ResetFilter_t

```
typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_ResetFilter_t
```

Definition at line 1070 of file cfe_evs_msg.h.

13.57.2.18 CFE_EVS_SetFilter_t

```
typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_SetFilter_t
```

Definition at line 1122 of file cfe_evs_msg.h.

13.57.2.19 CFE_EVS_TlmPkt_t

```
typedef CFE_EVS_HousekeepingTlm_t CFE_EVS_TlmPkt_t
```

Definition at line 1247 of file cfe_evs_msg.h.

13.58 cfe/fsw/cfe-core/src/inc/cfe_fs.h File Reference

```
#include "cfe_fs_extern_typedefs.h"
#include "common_types.h"
#include "cfe_time.h"
```

Macros

- #define CFE_FS_ES_ERLOG_SUBTYPE CFE_FS_SubType_ES_ERLOG
- #define CFE_FS_ES_SYSLOG_SUBTYPE CFE_FS_SubType_ES_SYSLOG
- #define CFE_FS_ES_QUERYALL_SUBTYPE CFE_FS_SubType_ES_QUERYALL
- #define CFE_FS_ES_PERFDATA_SUBTYPE CFE_FS_SubType_ES_PERFDATA
- #define CFE_FS_ES_SHELL_SUBTYPE CFE_FS_SubType_ES_SHELL
- #define CFE_FS_ES_CDS_REG_SUBTYPE CFE_FS_SubType_ES_CDS_REG
- #define CFE_FS_TBL_REG_SUBTYPE CFE_FS_SubType_TBL_REG
- #define CFE_FS_TBL_IMG_SUBTYPE CFE_FS_SubType_TBL_IMG
- #define CFE_FS_EVS_APPDATA_SUBTYPE CFE_FS_SubType_EVS_APPDATA
- #define CFE_FS_EVS_EVENTLOG_SUBTYPE CFE_FS_SubType_EVS_EVENTLOG
- #define CFE_FS_SB_PIPEDATA_SUBTYPE CFE_FS_SubType_SB_PIPEDATA
- #define CFE_FS_SB_ROUTEDATA_SUBTYPE CFE_FS_SubType_SB_ROUTEDATA
- #define CFE_FS_SB_MAPDATA_SUBTYPE CFE_FS_SubType_SB_MAPDATA
- #define CFE_FS_ES_QUERYALLTASKS_SUBTYPE CFE_FS_SubType_ES_QUERYALLTASKS

Functions

- int32 CFE_FS_ReadHeader (CFE_FS_Header_t ∗Hdr, int32 FileDes)

  Read the contents of the Standard cFE File Header.
- void CFE_FS_InitHeader (CFE_FS_Header_t ∗Hdr, const char ∗Description, uint32 SubType)

  Initializes the contents of the Standard cFE File Header.
- int32 CFE_FS_WriteHeader (int32 FileDes, CFE_FS_Header_t ∗Hdr)

  Write the specified Standard cFE File Header to the specified file.
- int32 CFE_FS_SetTimestamp (int32 FileDes, CFE_TIME_SysTime_t NewTimestamp)

  Modifies the Time Stamp field in the Standard cFE File Header for the specified file.
- bool CFE_FS_IsGzFile (const char ∗FileName)

  Determines if a file is a Gzip/compressed file.
- int32 CFE_FS_ExtractFilenameFromPath (const char ∗OriginalPath, char ∗FileNameOnly)

  Extracts the filename from a unix style path and filename string.
- int32 CFE_FS_Decompress (const char ∗SourceFile, const char ∗DestinationFile)

  Decompresses the source file to the destination file.
- int32 CFE_FS_GetUncompressedFile (char ∗OutputNameBuffer, uint32 OutputNameBufferSize, const char ∗GzipFileName, const char ∗TempDir)

  Decompresses the source file to a temporary file created in the temp dir.

13.58.1 Macro Definition Documentation

13.58.1.1 CFE_FS_ES_CDS_REG_SUBTYPE

```
#define CFE_FS_ES_CDS_REG_SUBTYPE CFE_FS_SubType_ES_CDS_REG
```

Definition at line 61 of file cfe_fs.h.

13.58.1.2 CFE_FS_ES_ERLOG_SUBTYPE

```
#define CFE_FS_ES_ERLOG_SUBTYPE CFE_FS_SubType_ES_ERLOG
```

Definition at line 56 of file cfe_fs.h.

13.58.1.3 CFE_FS_ES_PERFDATA_SUBTYPE

```
#define CFE_FS_ES_PERFDATA_SUBTYPE CFE_FS_SubType_ES_PERFDATA
```

Definition at line 59 of file cfe_fs.h.

13.58.1.4 CFE_FS_ES_QUERYALL_SUBTYPE

```
#define CFE_FS_ES_QUERYALL_SUBTYPE CFE_FS_SubType_ES_QUERYALL
```

Definition at line 58 of file cfe_fs.h.

13.58.1.5 CFE_FS_ES_QUERYALLTASKS_SUBTYPE

```
#define CFE_FS_ES_QUERYALLTASKS_SUBTYPE CFE_FS_SubType_ES_QUERYALLTASKS
```

Definition at line 69 of file cfe_fs.h.

13.58.1.6 CFE_FS_ES_SHELL_SUBTYPE

```
#define CFE_FS_ES_SHELL_SUBTYPE CFE_FS_SubType_ES_SHELL
```

Definition at line 60 of file cfe_fs.h.

13.58.1.7 CFE_FS_ES_SYSLOG_SUBTYPE

```
#define CFE_FS_ES_SYSLOG_SUBTYPE CFE_FS_SubType_ES_SYSLOG
```

Definition at line 57 of file cfe_fs.h.

13.58.1.8 CFE_FS_EVS_APPDATA_SUBTYPE

```
#define CFE_FS_EVS_APPDATA_SUBTYPE CFE_FS_SubType_EVS_APPDATA
```

Definition at line 64 of file cfe_fs.h.

13.58.1.9 CFE_FS_EVS_EVENTLOG_SUBTYPE

```
#define CFE_FS_EVS_EVENTLOG_SUBTYPE CFE_FS_SubType_EVS_EVENTLOG
```

Definition at line 65 of file cfe_fs.h.

13.58.1.10 CFE_FS_SB_MAPDATA_SUBTYPE

```
#define CFE_FS_SB_MAPDATA_SUBTYPE CFE_FS_SubType_SB_MAPDATA
```

Definition at line 68 of file cfe_fs.h.

13.58.1.11 CFE_FS_SB_PIPEDATA_SUBTYPE

```
#define CFE_FS_SB_PIPEDATA_SUBTYPE CFE_FS_SubType_SB_PIPEDATA
```

Definition at line 66 of file cfe_fs.h.

13.58.1.12 CFE_FS_SB_ROUTEDATA_SUBTYPE

```
#define CFE_FS_SB_ROUTEDATA_SUBTYPE CFE_FS_SubType_SB_ROUTEDATA
```

Definition at line 67 of file cfe_fs.h.

13.58.1.13 CFE_FS_TBL_IMG_SUBTYPE

```
#define CFE_FS_TBL_IMG_SUBTYPE CFE_FS_SubType_TBL_IMG
```

Definition at line 63 of file cfe_fs.h.

13.58.1.14 CFE_FS_TBL_REG_SUBTYPE

```
#define CFE_FS_TBL_REG_SUBTYPE CFE_FS_SubType_TBL_REG
```

Definition at line 62 of file cfe_fs.h.

13.58.2 Function Documentation

13.58.2.1 CFE_FS_Decompress()

```
int32 CFE_FS_Decompress (
            const char * SourceFile,
            const char * DestinationFile )
```

Description

This API will decompress the source file to the file specified by the destination file. The file must be compressed using the "gzip" utility. This utility is available on most unix workstations, Mac OS X, Cygwin, and MinGW for Windows. More information can be found at http://www.gzip.org/

Assumptions, External Events, and Notes:

1. The paths and filenames used here are cfe compliant file names.
2. The source file is compressed with the "gzip" utility.
3. The destination file does not exist, or can be overwritten.

Parameters

in	*SourceFile*	The "gzipped" file to decompress.
out	*DestinationFile*	The path/filename to write the decompressed or "gunzipped" file to.

Returns

CFE_SUCCESS if the file was decompressed sucessfully.

See also

Referenced by CFE_ES_AppCreate().

13.58.2.2 CFE_FS_ExtractFilenameFromPath()

```
int32 CFE_FS_ExtractFilenameFromPath (
            const char * OriginalPath,
            char * FileNameOnly )
```

Description

This API will take the original unix path/filename combination and extract the base filename. Example: Given the path/filename : "/cf/apps/myapp.o.gz" this function will return the filename: "myapp.o.gz".

Assumptions, External Events, and Notes:

1. The paths and filenames used here are the standard unix style filenames separated by "/" characters.
2. The extracted filename is no longer than OS_MAX_PATH_LEN

Parameters

in	*OriginalPath*	The original path.
out	*FileNameOnly*	The filename that is extracted from the path.

Returns

CFE_SUCCESS if the filename was extracted from the path

See also

Referenced by CFE_ES_AppCreate().

13.58.2.3 CFE_FS_GetUncompressedFile()

```
int32 CFE_FS_GetUncompressedFile (
            char * OutputNameBuffer,
            uint32 OutputNameBufferSize,
            const char * GzipFileName,
            const char * TempDir )
```

Description

This is a wrapper around the CFE_FS_Decompress() function that formulates a temporary file name based on the gzip file name, saving the caller from needing to do this. The temporary file name is created in the given temp directory.

Assumptions, External Events, and Notes:

The name passed in as "GzipFileName" is not checked again, it is assumed to have passed the criteria in CFE_FS_IsGzFile(). If this is not true then the conversion to a temporary file name may produce incorrect results.

Parameters

in	*OutputNameBuffer*	A caller-supplied buffer for storing the temp file name
in	*OutputNameBufferSize*	The size of OutputNameBuffer
in	*GzipFileName*	The "gzipped" file to decompress.
in	*TempDir*	The directory in which the temporary file should be created

Returns

CFE_SUCCESS if the file was decompressed sucessfully.

See also

Referenced by CFE_ES_LoadLibrary().

13.58.2.4 CFE_FS_InitHeader()

```
void CFE_FS_InitHeader (
            CFE_FS_Header_t * Hdr,
            const char * Description,
            uint32 SubType )
```

Description

This API will clear the specified CFE_FS_Header_t variable and initialize the description field with the specified value

Parameters

in	*Hdr*	Pointer to a variable of type CFE_FS_Header_t that will be cleared and initialized
in	∗*Description*	Initializes Header's Description
in	*SubType*	Initializes Header's SubType

See also

CFE_FS_WriteHeader

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_PerfLogDump(), CFE_ES_↩ QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_SysLogDump(), CFE_EVS_WriteAppDataFileCmd(), CFE↩ _EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.58.2.5 CFE_FS_IsGzFile()

```
bool CFE_FS_IsGzFile (
            const char * FileName )
```

Description

This API will check the filename and return true if the file is a gzip file. The check is currently based on the filename, so the zipped files should use the ".gz" extention.

Assumptions, External Events, and Notes:

1. A gzipped file will use the ".gz" filename extention.

Parameters

in	*FileName*	The name of the file.

Returns

true if the file has the ".gz" extention and false otherwise.

See also

Referenced by CFE_ES_AppCreate(), and CFE_ES_LoadLibrary().

13.58.2.6 CFE_FS_ReadHeader()

```
int32 CFE_FS_ReadHeader (
            CFE_FS_Header_t * Hdr,
            int32 FileDes )
```

Description

This API will fill the specified CFE_FS_Header_t variable with the contents of the Standard cFE File Header of the file identified by the given File Descriptor.

Assumptions, External Events, and Notes:

1. The File has already been successfully opened using OS_open and the caller has a legitimate File Descriptor.

Parameters

in	*FileDes*	File Descriptor obtained from a previous call to OS_open that is associated with the file whose header is to be read.
in	*Hdr*	Pointer to a variable of type CFE_FS_Header_t that will be filled with the contents of the Standard cFE File Header.
out	**Hdr*	Contents of the Standard cFE File Header for the specified file.

Any of the return codes specified for OS_lseek or OS_read

Returns

See also

CFE_FS_WriteHeader

13.58.2.7 CFE_FS_SetTimestamp()

```
int32 CFE_FS_SetTimestamp (
            int32 FileDes,
            CFE_TIME_SysTime_t NewTimestamp )
```

Description

This API will modify the timestamp found in the Standard cFE File Header of the specified file. The timestamp will be replaced with the time specified by the caller.

Assumptions, External Events, and Notes:

1. The File has already been successfully opened using OS_open and the caller has a legitimate File Descriptor.
2. The `NewTimestamp` field has been filled appropriately by the Application.

Parameters

in	*FileDes*	File Descriptor obtained from a previous call to OS_open that is associated with the file whose header is to be read.
in	*NewTimestamp*	A CFE_TIME_SysTime_t data structure containing the desired time to be put into the file's Standard cFE File Header.

Any of the return codes specified for OS_lseek or OS_write

Returns

See also

13.58.2.8 CFE_FS_WriteHeader()

```
int32 CFE_FS_WriteHeader (
            int32 FileDes,
            CFE_FS_Header_t * Hdr )
```

Description

This API will output the specified CFE_FS_Header_t variable, with some fields automatically updated, to the specified file as the Standard cFE File Header. This API will automatically populate the following fields in the specified CFE_FS_Header_t:

1. `ContentType` - Filled with 0x63464531 ('cFE1')
2. `Length` - Filled with the sizeof(CFE_FS_Header_t)
3. `SpacecraftID` - Filled with the Spacecraft ID
4. `ProcessorID` - Filled with the Processor ID
5. `ApplicationID` - Filled with the Application ID
6. `TimeSeconds` - Filled with the Time, in seconds, as obtained by CFE_TIME_GetTime
7. `TimeSubSeconds` - Filled with the Time, subseconds, as obtained by CFE_TIME_GetTime

Assumptions, External Events, and Notes:

1. The File has already been successfully opened using OS_open and the caller has a legitimate File Descriptor.
2. The `SubType` field has been filled appropriately by the Application.
3. The `Description` field has been filled appropriately by the Application.

Parameters

in	*FileDes*	File Descriptor obtained from a previous call to OS_open that is associated with the file whose header is to be read.
in	*Hdr*	Pointer to a variable of type CFE_FS_Header_t that will be filled with the contents of the Standard cFE File Header.
out	**Hdr*	Contents of the Standard cFE File Header for the specified file.

Any of the return codes specified for OS_lseek or OS_write

Returns

See also

CFE_FS_ReadHeader

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_PerfLogDump(), CFE_ES_↩ QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_SysLogDump(), CFE_EVS_WriteAppDataFileCmd(), CFE↩ _EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.59 cfe/fsw/cfe-core/src/inc/cfe_fs_extern_typedefs.h File Reference

```
#include "common_types.h"
```

Data Structures

- struct CFE_FS_Header_t

  Standard cFE File header structure definition.

Macros

- #define CFE_FS_HDR_DESC_MAX_LEN 32

  Max length of description field in a standard cFE File Header.
- #define CFE_FS_FILE_CONTENT_ID 0x63464531

  Magic Number for cFE compliant files (= 'cFE1')

Typedefs

- typedef uint32 CFE_FS_SubType_Enum_t

  Content descriptor for File Headers.

Enumerations

- enum CFE_FS_SubType {
  CFE_FS_SubType_ES_ERLOG = 1, CFE_FS_SubType_ES_SYSLOG = 2, CFE_FS_SubType_ES_QUERYALL = 3, CFE_FS_SubType_ES_PERFDATA = 4,
  CFE_FS_SubType_ES_SHELL = 5, CFE_FS_SubType_ES_CDS_REG = 6, CFE_FS_SubType_TBL_REG = 9, CFE_FS_SubType_TBL_IMG = 8,
  CFE_FS_SubType_EVS_APPDATA = 15, CFE_FS_SubType_EVS_EVENTLOG = 16, CFE_FS_SubType_SB_PIPEDATA = 20, CFE_FS_SubType_SB_ROUTEDATA = 21,
  CFE_FS_SubType_SB_MAPDATA = 22, CFE_FS_SubType_ES_QUERYALLTASKS = 23 }

  Label definitions associated with CFE_FS_SubType_Enum_t.

13.59.1 Macro Definition Documentation

13.59.1.1 CFE_FS_FILE_CONTENT_ID

```
#define CFE_FS_FILE_CONTENT_ID 0x63464531
```

Definition at line 47 of file cfe_fs_extern_typedefs.h.

13.59.1.2 CFE_FS_HDR_DESC_MAX_LEN

```
#define CFE_FS_HDR_DESC_MAX_LEN 32
```

Definition at line 45 of file cfe_fs_extern_typedefs.h.

13.59.2 Typedef Documentation

13.59.2.1 CFE_FS_SubType_Enum_t

```
typedef uint32 CFE_FS_SubType_Enum_t
```

See also

enum CFE_FS_SubType

Definition at line 217 of file cfe_fs_extern_typedefs.h.

13.59.3 Enumeration Type Documentation

13.59.3.1 CFE_FS_SubType

```
enum CFE_FS_SubType
```

Enumerator

CFE_FS_SubType_ES_ERLOG	Executive Services Exception/Reset Log Type. Executive Services Exception/Reset Log File which is generated in response to a $sc_$cpu_ES_WriteERLog2File command.
CFE_FS_SubType_ES_SYSLOG	Executive Services System Log Type. Executive Services System Log File which is generated in response to a $sc_$cpu_ES_WriteSysLog2File command.
CFE_FS_SubType_ES_QUERYALL	Executive Services Information on All Applications File. Executive Services Information on All Applications File which is generated in response to a $sc_$cpu_ES_WriteAppInfo2File command.
CFE_FS_SubType_ES_PERFDATA	Executive Services Performance Data File. Executive Services Performance Analyzer Data File which is generated in response to a $sc_$cpu_ES_StopLAData command.
CFE_FS_SubType_ES_SHELL	Executive Services Shell Response File. Executive Services Shell Response Data File which is generated in response to a $sc_$cpu$_ES_Shell command.
CFE_FS_SubType_ES_CDS_REG	Executive Services Critical Data Store Registry Dump File. Executive Services Critical Data Store Registry Dump File which is generated in response to a $sc_$cpu_ES_WriteCDS2File command.
CFE_FS_SubType_TBL_REG	Table Services Registry Dump File. Table Services Registry Dump File which is generated in response to a $sc_$cpu_TBL_WriteReg2File command.
CFE_FS_SubType_TBL_IMG	Table Services Table Image File. Table Services Table Image File which is generated either on the ground or in response to a $sc_$cpu_TBL_DUMP command.
CFE_FS_SubType_EVS_APPDATA	Event Services Application Data Dump File. Event Services Application Data Dump File which is generated in response to a $sc_$cpu_EVS_WriteAppData2File command.
CFE_FS_SubType_EVS_EVENTLOG	Event Services Local Event Log Dump File. Event Services Local Event Log Dump File which is generated in response to a $sc_$cpu_EVS_WriteLog2File command.
CFE_FS_SubType_SB_PIPEDATA	Software Bus Pipe Data Dump File. Software Bus Pipe Data Dump File which is generated in response to a $sc_$cpu_SB_WritePipe2File command.
CFE_FS_SubType_SB_ROUTEDATA	Software Bus Message Routing Data Dump File. Software Bus Message Routing Data Dump File which is generated in response to a $sc_$cpu_SB_WriteRouting2File command.
CFE_FS_SubType_SB_MAPDATA	Software Bus Message Mapping Data Dump File. Software Bus Message Mapping Data Dump File which is generated in response to a $sc_$cpu_SB_WriteMap2File command.
CFE_FS_SubType_ES_QUERYALLTASKS	Executive Services Query All Tasks Data File. Executive Services Query All Tasks Data File which is generated in response to a $sc_$cpu_ES_WriteTaskInfo2File command.

Definition at line 54 of file cfe_fs_extern_typedefs.h.

13.60 cfe/fsw/cfe-core/src/inc/cfe_sb.h File Reference

```
#include "cfe_sb_extern_typedefs.h"
#include "osconfig.h"
#include "cfe_psp.h"
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "ccsds.h"
#include "cfe_time.h"
```

Data Structures

- union CFE_SB_Msg_t

  Generic Software Bus Message Type Definition.
- struct CFE_SB_Qos_t
- struct CFE_SB_SenderId_t

Macros

- #define CFE_SB_POLL 0

  Option used with CFE_SB_RcvMsg to request immediate pipe status.
- #define CFE_SB_PEND_FOREVER -1

  Option used with CFE_SB_RcvMsg to force a wait for next message.
- #define CFE_SB_SUB_ENTRIES_PER_PKT 20

  Configuration parameter used by SBN App.
- #define CFE_SB_SUBSCRIPTION 0

  Subtype specifier used in CFE_SB_SingleSubscriptionTlm_t by SBN App.
- #define CFE_SB_UNSUBSCRIPTION 1

  Subtype specified used in CFE_SB_SingleSubscriptionTlm_t by SBN App.
- #define CFE_SB_INVALID_MSG_ID 0xFFFF

  Initializer for CFE_SB_MsgId_t values that will not match any real MsgId.
- #define CFE_BIT(x) (1 << (x))

  Places a one at bit positions 0 - 31.
- #define CFE_SET(i, x) ((i) |= CFE_BIT(x))

  Sets bit x of i.
- #define CFE_CLR(i, x) ((i) &= ~CFE_BIT(x))

  Clears bit x of i.
- #define CFE_TST(i, x) (((i) & CFE_BIT(x)) != 0)

  true(non zero) if bit x of i is set
- #define CFE_SB_SET_MEMADDR(msgdst, src) msgdst = (cpuaddr)src
- #define CFE_SB_GET_MEMADDR(msgsrc) (cpuaddr)msgsrc
- #define CFE_SB_PIPEOPTS_IGNOREMINE 0x00000001

  Messages sent by the app that owns this pipe will not be sent to this pipe.
- #define CFE_SB_CMD_HDR_SIZE (sizeof(CFE_SB_CmdHdr_t))

  Size of CFE_SB_CmdHdr_t in bytes.
- #define CFE_SB_TLM_HDR_SIZE (sizeof(CFE_SB_TlmHdr_t))

  Size of CFE_SB_TlmHdr_t in bytes.

Typedefs

- typedef CCSDS_CommandPacket_t CFE_SB_CmdHdr_t

  Generic Software Bus Command Header Type Definition.
- typedef CCSDS_TelemetryPacket_t CFE_SB_TlmHdr_t

  Generic Software Bus Telemetry Header Type Definition.
- typedef uint32 CFE_SB_TimeOut_t

  CFE_SB_TimeOut_t to primitive type definition.
- typedef uint8 CFE_SB_PipeId_t

  CFE_SB_MsgPtr_t defined as a pointer to an SB Message.
- typedef CFE_SB_Msg_t * CFE_SB_MsgPtr_t

  CFE_SB_MsgPayloadPtr_t defined as an opaque pointer to a message Payload portion.
- typedef uint8 * CFE_SB_MsgPayloadPtr_t

  CFE_SB_ZeroCopyId_t to primitive type definition.
- typedef cpuaddr CFE_SB_ZeroCopyHandle_t

  Quality Of Service Type Definition.

Functions

- int32 CFE_SB_CreatePipe (CFE_SB_PipeId_t *PipeIdPtr, uint16 Depth, const char *PipeName)

  Creates a new software bus pipe.
- int32 CFE_SB_DeletePipe (CFE_SB_PipeId_t PipeId)

  Delete a software bus pipe.
- int32 CFE_SB_SetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 Opts)

  Set options on a pipe.
- int32 CFE_SB_GetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 *OptPtr)

  Get options on a pipe.
- int32 CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_PipeId_t PipeId)

  Get the pipe name for a given id.
- int32 CFE_SB_GetPipeIdByName (CFE_SB_PipeId_t *PipeIdPtr, const char *PipeName)

  Get pipe id by pipe name.
- int32 CFE_SB_SubscribeEx (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, CFE_SB_Qos_t Quality, uint16 MsgLim)

  Subscribe to a message on the software bus.
- int32 CFE_SB_Subscribe (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId)

  Subscribe to a message on the software bus with default parameters.
- int32 CFE_SB_SubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, uint16 MsgLim)

  Subscribe to a message while keeping the request local to a cpu.
- int32 CFE_SB_Unsubscribe (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId)

  Remove a subscription to a message on the software bus.
- int32 CFE_SB_UnsubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId)

  Remove a subscription to a message on the software bus on the current CPU.
- int32 CFE_SB_SendMsg (CFE_SB_Msg_t *MsgPtr)

  Send a software bus message.
- int32 CFE_SB_PassMsg (CFE_SB_Msg_t *MsgPtr)

  Passes a software bus message.
- int32 CFE_SB_RcvMsg (CFE_SB_MsgPtr_t *BufPtr, CFE_SB_PipeId_t PipeId, int32 TimeOut)

Receive a message from a software bus pipe.

- uint32 CFE_SB_GetLastSenderId (CFE_SB_SenderId_t **Ptr, CFE_SB_PipeId_t PipeId)

  Retrieve the application Info of the sender for the last message.
- CFE_SB_Msg_t * CFE_SB_ZeroCopyGetPtr (uint16 MsgSize, CFE_SB_ZeroCopyHandle_t *BufferHandle)

  Get a buffer pointer to use for "zero copy" SB sends.
- int32 CFE_SB_ZeroCopyReleasePtr (CFE_SB_Msg_t *Ptr2Release, CFE_SB_ZeroCopyHandle_t BufferHandle)

  Release an unused "zero copy" buffer pointer.
- int32 CFE_SB_ZeroCopySend (CFE_SB_Msg_t *MsgPtr, CFE_SB_ZeroCopyHandle_t BufferHandle)

  Send an SB message in "zero copy" mode.
- int32 CFE_SB_ZeroCopyPass (CFE_SB_Msg_t *MsgPtr, CFE_SB_ZeroCopyHandle_t BufferHandle)

  Pass an SB message in "zero copy" mode.
- void CFE_SB_InitMsg (void *MsgPtr, CFE_SB_MsgId_t MsgId, uint16 Length, bool Clear)

  Initialize a buffer for a software bus message.
- void * CFE_SB_GetUserData (CFE_SB_MsgPtr_t MsgPtr)

  Get a pointer to the user data portion of a software bus message.
- CFE_SB_MsgId_t CFE_SB_GetMsgId (const CFE_SB_Msg_t *MsgPtr)

  Get the message ID of a software bus message.
- void CFE_SB_SetMsgId (CFE_SB_MsgPtr_t MsgPtr, CFE_SB_MsgId_t MsgId)

  Sets the message ID of a software bus message.
- uint16 CFE_SB_GetUserDataLength (const CFE_SB_Msg_t *MsgPtr)

  Gets the length of user data in a software bus message.
- void CFE_SB_SetUserDataLength (CFE_SB_MsgPtr_t MsgPtr, uint16 DataLength)

  Sets the length of user data in a software bus message.
- uint16 CFE_SB_GetTotalMsgLength (const CFE_SB_Msg_t *MsgPtr)

  Gets the total length of a software bus message.
- void CFE_SB_SetTotalMsgLength (CFE_SB_MsgPtr_t MsgPtr, uint16 TotalLength)

  Sets the total length of a software bus message.
- CFE_TIME_SysTime_t CFE_SB_GetMsgTime (CFE_SB_MsgPtr_t MsgPtr)

  Gets the time field from a software bus message.
- int32 CFE_SB_SetMsgTime (CFE_SB_MsgPtr_t MsgPtr, CFE_TIME_SysTime_t Time)

  Sets the time field in a software bus message.
- void CFE_SB_TimeStampMsg (CFE_SB_MsgPtr_t MsgPtr)

  Sets the time field in a software bus message with the current spacecraft time.
- uint16 CFE_SB_GetCmdCode (CFE_SB_MsgPtr_t MsgPtr)

  Gets the command code field from a software bus message.
- int32 CFE_SB_SetCmdCode (CFE_SB_MsgPtr_t MsgPtr, uint16 CmdCode)

  Sets the command code field in a software bus message.
- uint16 CFE_SB_GetChecksum (CFE_SB_MsgPtr_t MsgPtr)

  Gets the checksum field from a software bus message.
- void CFE_SB_GenerateChecksum (CFE_SB_MsgPtr_t MsgPtr)

  Calculates and sets the checksum of a software bus message.
- bool CFE_SB_ValidateChecksum (CFE_SB_MsgPtr_t MsgPtr)

  Validates the checksum of a software bus message.
- int32 CFE_SB_MessageStringGet (char *DestStringPtr, const char *SourceStringPtr, const char *DefaultString, uint32 DestMaxSize, uint32 SourceMaxSize)
- int32 CFE_SB_MessageStringSet (char *DestStringPtr, const char *SourceStringPtr, uint32 DestMaxSize, uint32 SourceMaxSize)

- static bool CFE_SB_MsgId_Equal (CFE_SB_MsgId_t MsgId1, CFE_SB_MsgId_t MsgId2)

  Identifies whether a two CFE_SB_MsgId_t values are equal.
- static CFE_SB_MsgId_Atom_t CFE_SB_MsgIdToValue (CFE_SB_MsgId_t MsgId)

  Converts a CFE_SB_MsgId_t to a normal integer.
- static CFE_SB_MsgId_t CFE_SB_ValueToMsgId (CFE_SB_MsgId_Atom_t MsgIdValue)

  Converts a normal integer into a CFE_SB_MsgId_t.

Variables

- CFE_SB_Qos_t CFE_SB_Default_Qos

  Defines a default priority and reliabilty for off-board routing.

13.60.1 Macro Definition Documentation

13.60.1.1 CFE_BIT

```
#define CFE_BIT(
              x ) (1 << (x))
```

Definition at line 61 of file cfe_sb.h.

13.60.1.2 CFE_CLR

```
#define CFE_CLR(
              i,
              x ) ((i) &= ~CFE_BIT(x))
```

Definition at line 63 of file cfe_sb.h.

Referenced by CFE_SB_FinishSendEvent().

13.60.1.3 CFE_SB_CMD_HDR_SIZE

```
#define CFE_SB_CMD_HDR_SIZE (sizeof(CFE_SB_CmdHdr_t))
```

Definition at line 104 of file cfe_sb.h.

Referenced by CFE_SB_MsgHdrSize().

13.60.1.4 CFE_SB_GET_MEMADDR

```
#define CFE_SB_GET_MEMADDR(
              msgsrc ) (cpuaddr)msgsrc
```

Macro that should be used to get memory addresses from software bus messages. This is the inverse operation of CFE_SB_SET_MEMADDR.

Definition at line 78 of file cfe_sb.h.

Referenced by CFE_ES_SendMemPoolStatsCmd().

13.60.1.5 CFE_SB_INVALID_MSG_ID

```
#define CFE_SB_INVALID_MSG_ID 0xFFFF
```

Definition at line 56 of file cfe_sb.h.

Referenced by CFE_SB_InitRoutingTbl(), and CFE_SB_IsValidMsgId().

13.60.1.6 CFE_SB_PEND_FOREVER

```
#define CFE_SB_PEND_FOREVER -1
```

Definition at line 51 of file cfe_sb.h.

Referenced by CFE_EVS_TaskMain(), CFE_SB_ReadQueue(), and CFE_SB_TaskMain().

13.60.1.7 CFE_SB_PIPEOPTS_IGNOREMINE

```
#define CFE_SB_PIPEOPTS_IGNOREMINE 0x00000001
```

Definition at line 83 of file cfe_sb.h.

Referenced by CFE_SB_SendMsgFull().

13.60.1.8 CFE_SB_POLL

```
#define CFE_SB_POLL 0
```

Definition at line 50 of file cfe_sb.h.

Referenced by CFE_SB_DeletePipeFull(), and CFE_SB_ReadQueue().

13.60.1.9 CFE_SB_SET_MEMADDR

```
#define CFE_SB_SET_MEMADDR(
             msgdst,
             src ) msgdst = (cpuaddr)src
```

Macro that should be used to set memory addresses within software bus messages. For now this does a straight copy, but in a future revision this may translate the raw memory address into a "safe" integer value. This is particularly important if the message is to be sent off this CPU.

Definition at line 72 of file cfe_sb.h.

Referenced by CFE_ES_GetAppInfoInternal(), CFE_ES_SendMemPoolStatsCmd(), and CFE_SB_AppInit().

13.60.1.10 CFE_SB_SUB_ENTRIES_PER_PKT

```
#define CFE_SB_SUB_ENTRIES_PER_PKT 20
```

Definition at line 52 of file cfe_sb.h.

Referenced by CFE_SB_SendPrevSubsCmd().

13.60.1.11 CFE_SB_SUBSCRIPTION

```
#define CFE_SB_SUBSCRIPTION 0
```

Definition at line 53 of file cfe_sb.h.

Referenced by CFE_SB_SubscribeFull().

13.60.1.12 CFE_SB_TLM_HDR_SIZE

```
#define CFE_SB_TLM_HDR_SIZE (sizeof(CFE_SB_TlmHdr_t))
```

Definition at line 105 of file cfe_sb.h.

Referenced by CFE_SB_MsgHdrSize().

13.60.1.13 CFE_SB_UNSUBSCRIPTION

```
#define CFE_SB_UNSUBSCRIPTION 1
```

Definition at line 54 of file cfe_sb.h.

13.60.1.14 CFE_SET

```
#define CFE_SET(
              i,
              x ) ((i) |= CFE_BIT(x))
```

Definition at line 62 of file cfe_sb.h.

Referenced by CFE_SB_RequestToSendEvent().

13.60.1.15 CFE_TST

```
#define CFE_TST(
              i,
              x ) (((i) & CFE_BIT(x)) != 0)
```

Definition at line 64 of file cfe_sb.h.

Referenced by CFE_SB_GetPktType(), and CFE_SB_RequestToSendEvent().

13.60.2 Typedef Documentation

13.60.2.1 CFE_SB_CmdHdr_t

```
typedef CCSDS_CommandPacket_t CFE_SB_CmdHdr_t
```

Definition at line 99 of file cfe_sb.h.

13.60.2.2 CFE_SB_MsgPayloadPtr_t

```
typedef uint8* CFE_SB_MsgPayloadPtr_t
```

Software Zero Copy handle used in many SB APIs

Definition at line 127 of file cfe_sb.h.

13.60.2.3 CFE_SB_MsgPtr_t

```
typedef CFE_SB_Msg_t* CFE_SB_MsgPtr_t
```

Definition at line 124 of file cfe_sb.h.

13.60.2.4 CFE_SB_PipeId_t

```
typedef uint8 CFE_SB_PipeId_t
```

Definition at line 121 of file cfe_sb.h.

13.60.2.5 CFE_SB_TimeOut_t

```
typedef uint32 CFE_SB_TimeOut_t
```

<Internally used by SB in the CFE_SB_RcvMsg API. Translated from the input parmater named TimeOut which specifies the maximum time in milliseconds that the caller wants to wait for a message.CFE_SB_PipeId_t to primitive type definition

Software Bus pipe identifier used in many SB APIs

Definition at line 115 of file cfe_sb.h.

13.60.2.6 CFE_SB_TlmHdr_t

```
typedef CCSDS_TelemetryPacket_t CFE_SB_TlmHdr_t
```

Definition at line 102 of file cfe_sb.h.

13.60.2.7 CFE_SB_ZeroCopyHandle_t

```
typedef cpuaddr CFE_SB_ZeroCopyHandle_t
```

Currently an unused parameter in CFE_SB_SubscribeEx Intended to be used for interprocessor communication only

Definition at line 133 of file cfe_sb.h.

13.60.3 Function Documentation

13.60.3.1 CFE_SB_CreatePipe()

```
int32 CFE_SB_CreatePipe (
            CFE_SB_PipeId_t * PipeIdPtr,
            uint16 Depth,
            const char * PipeName )
```

Description

This routine creates and initializes an input pipe that the calling application can use to receive software bus messages. By default, no messages are routed to the new pipe. So, the application must use CFE_SB_Subscribe() to specify which messages it wants to receive on this pipe.

Assumptions, External Events, and Notes:

None

Parameters

in	*PipeIdPtr*	A pointer to a variable of type CFE_SB_PipeId_t, which will be filled in with the pipe ID information by the CFE_SB_CreatePipe routine.
in	*Depth*	The maximum number of messages that will be allowed on this pipe at one time.
in	*PipeName*	A string to be used to identify this pipe in error messages and routing information telemetry. The string must be no longer than OS_MAX_API_NAME. Longer strings will be truncated.
out	∗*PipeIdPtr*	The identifier for the created pipe.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MAX_PIPES_MET	This error code will be returned from CFE_SB_CreatePipe when the SB cannot accomodate the request to create a pipe because the maximum number of pipes (CFE_PLATFORM_SB_MAX_PIPES) are in use. This configuration parameter is defined in the cfe_platform_cfg.h file.
CFE_SB_PIPE_CR_ERR	The maximum number of queues(OS_MAX_QUEUES) are in use. Or possibly a lower level problem with creating the underlying queue has occurred such as a lack of memory. If the latter is the problem, the status code displayed in the event must be tracked.

Returns

See also

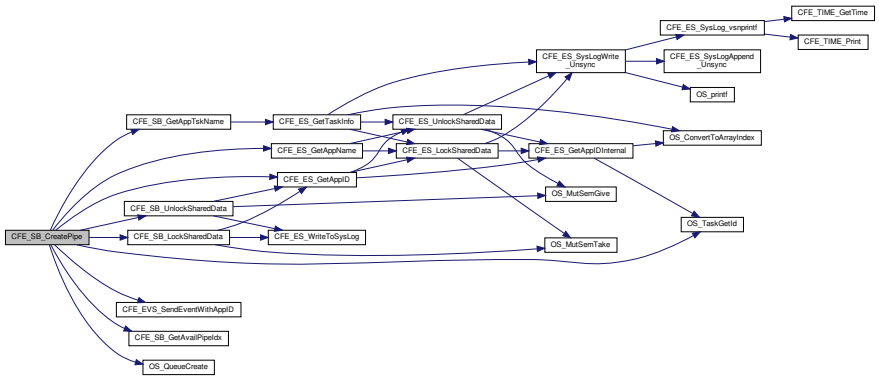
CFE_SB_DeletePipe CFE_SB_GetPipeOpts CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

Definition at line 104 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_SB_PipeD_t::AppName, CFE_ES_GetAppID(), CFE_ES_GetAppName(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_PIPE_DEPTH, CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_CR_PIPE_BAD_ARG_EID, CFE_SB_CR_PIPE_ERR_EID, CFE_SB_CR_PIPE_NAME_TAKEN_EID, CFE_SB_CR_PIPE_NO_FREE_EID, CFE_SB_GetAppTskName(), CFE_SB_GetAvailPipeIdx(), CFE_SB_IN_USE, CFE_SB_INVALID_PIPE, CFE_SB_LockSharedData(), CFE_SB_MAX_PIPES_MET, CFE_SB_MAX_PIPES_MET_EID, CFE_SB_PIPE_ADDED_EID, CFE_SB_PIPE_CR_ERR, CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter, CFE_SB_PipeD_t::CurrentBuff, CFE_SB_PipeDepthStats_t::Depth, cfe_sb_t::HKTlmMsg, CFE_SB_PipeD_t::InUse, CFE_SB_PipeDepthStats_t::InUse, NULL, OS_ERR_NAME_TAKEN, OS_ERR_NO_FREE_IDS, OS_MAX_API_NAME, OS_QueueCreate(), OS_SUCCESS, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_PipeDepthStats_t::PeakInUse, CFE_SB_StatsTlm_Payload_t::PeakPipesInUse, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_PipeD_t::PipeId, CFE_SB_PipeDepthStats_t::PipeId, CFE_SB_StatsTlm_Payload_t::PipesInUse, cfe_sb_t::PipeTbl, CFE_SB_PipeD_t::QueueDepth, CFE_SB_PipeD_t::SendErrors, cfe_sb_t::StatTlmMsg, CFE_SB_PipeD_t::SysQueueId, and CFE_SB_PipeD_t::ToTrashBuff.

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), and CFE_SB_AppInit().

Here is the call graph for this function:



13.60.3.2 CFE_SB_DeletePipe()

```
int32 CFE_SB_DeletePipe (
            CFE_SB_PipeId_t PipeId )
```

Description

This routine deletes an input pipe and cleans up all data structures associated with the pipe. All subscriptions made for this pipe by calls to CFE_SB_Subscribe will be automatically removed from the SB routing tables. Any messages in the pipe will be discarded.

Applications should not call this routine for all of their SB pipes as part of their orderly shutdown process, as the pipe will be deleted by the support framework at the appropriate time.

Assumptions, External Events, and Notes:

None

Parameters

in	*PipeId*	The pipe ID (obtained previously from CFE_SB_CreatePipe) of the pipe to be deleted.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.

Returns

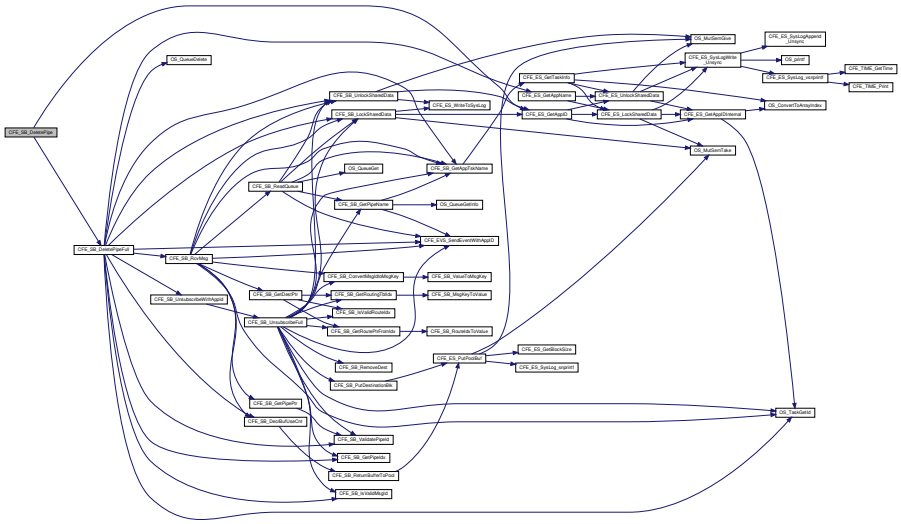
See also

CFE_SB_CreatePipe CFE_SB_GetPipeOpts CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

Definition at line 243 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), and CFE_SB_DeletePipeFull().

Here is the call graph for this function:



13.60.3.3 CFE_SB_GenerateChecksum()

```
void CFE_SB_GenerateChecksum (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine calculates the checksum of a software bus message according to an implementation-defined algorithm. Then, it sets the checksum field in the message with the calculated value. The contents and location of this field will depend on the underlying implementation of software bus messages. It may be a checksum, a CRC, or some other algorithm.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a checksum field, then this routine will do nothing.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

See also

CFE_SB_ValidateChecksum, CFE_SB_GetChecksum

Definition at line 550 of file cfe_sb_util.c.

References CCSDS_LoadCheckSum(), CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, and CFE_SB_Msg↩_t::Hdr.

Here is the call graph for this function:

13.60.3.4 CFE_SB_GetChecksum()

```
uint16 CFE_SB_GetChecksum (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine gets the checksum (or other message integrity check value) from a software bus message. The contents and location of this field will depend on the underlying implementation of software bus messages. It may be a checksum, a CRC, or some other algorithm. Users should not call this function as part of a message integrity check (call CFE_SB_ValidateChecksum instead).

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a checksum field, then this routine will return a zero.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The checksum included in the software bus message header (if present), otherwise, returns a checksum value of zero.

Returns

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize, CFE_SB_ValidateChecksum, CFE_SB_GenerateChecksum

Definition at line 512 of file cfe_sb_util.c.

References CCSDS_RD_CHECKSUM, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CFE_SB_Msg_t::Hdr, and CCSDS_CommandPacket_t::Sec.

13.60.3.5 CFE_SB_GetCmdCode()

```
uint16 CFE_SB_GetCmdCode (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine gets the command code from a software bus message (if SB messages are implemented as CCSDS packets, this will be the function code).

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a command code field, then this routine will return a zero.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The command code included in the software bus message header (if present). Otherwise, returns a command code value of zero.

Returns

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_SetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 443 of file cfe_sb_util.c.

References CCSDS_RD_FC, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CFE_SB_Msg_t::Hdr, and CCSDS_CommandPacket_t::Sec.

Referenced by CFE_ES_TaskPipe(), CFE_ES_VerifyCmdLength(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_VerifyCmdLength(), CFE_SB_ProcessCmdPipePkt(), and CFE_SB_VerifyCmdLength().

13.60.3.6 CFE_SB_GetLastSenderId()

```
uint32 CFE_SB_GetLastSenderId (
            CFE_SB_SenderId_t ** Ptr,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine can be used after a successful CFE_SB_RcvMsg call to find out which application sent the message that was received.

Assumptions, External Events, and Notes:

Note - If an error occurs in this API, the ∗Ptr value may be NULL or random. Therefore, it is recommended that the return code be tested for CFE_SUCCESS before reading the sender information.

Parameters

in	*Ptr*	A pointer to a local variable of type CFE_SB_SenderId_t. Typically a caller declares a ptr of type CFE_SB_SenderId_t (i.e. CFE_SB_SenderId_t ∗Ptr) then gives the address of that pointer (&Ptr) for this parameter. After a successful call to this API, ∗Ptr will point to the first byte of the CFE_SB_SenderId_t structure containing the sender information for the last message received on the given pipe. This should be used as a read-only pointer (in systems with an MMU, writes to this pointer may cause a memory protection fault). The ∗Ptr is valid only until the next call to CFE_SB_RcvMsg for the same pipe.
in	*PipeId*	The pipe ID of the pipe the message was taken from.

The sender's application ID	

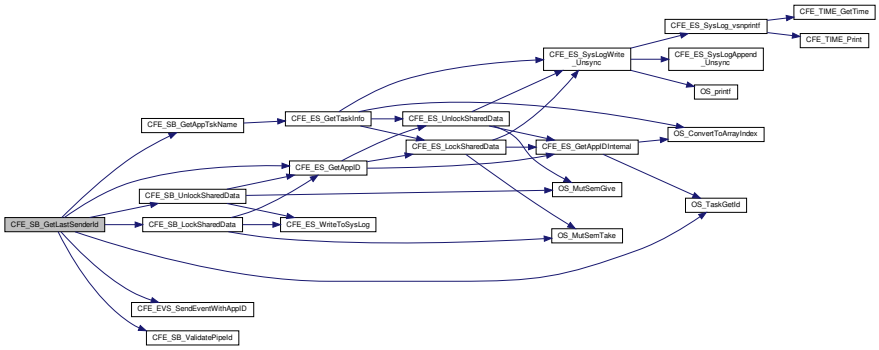
Returns

See also

Definition at line 1883 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_EVS_EventType_ERROR, CFE_↩ EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GLS_I↩ NV_CALLER_EID, CFE_SB_LockSharedData(), CFE_SB_LSTSNDER_ERR1_EID, CFE_SB_LSTSNDER_ERR2_E↩ ID, CFE_SB_UnlockSharedData(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_PipeD_t::CurrentBuff, NULL, OS_MAX_API_NAME, OS_TaskGetId(), and cfe_sb_t::PipeTbl.

Here is the call graph for this function:



13.60.3.7 CFE_SB_GetMsgId()

```
CFE_SB_MsgId_t CFE_SB_GetMsgId (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the message ID from a software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message.

The software bus Message ID from the message header.

Returns

See also

CFE_SB_GetUserData, CFE_SB_SetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 132 of file cfe_sb_msg_id_util.c.

References CCSDS_CMD, CCSDS_RD_APID, CCSDS_RD_SID, CCSDS_RD_SUBSYSTEM_ID, CCSDS_RD_TYPE, CFE_SB_CMD_MESSAGE_TYPE, CFE_SB_Msg_t::Hdr, and CFE_SB_Msg_t::SpacePacket.

Referenced by CFE_ES_TaskPipe(), CFE_ES_VerifyCmdLength(), CFE_EVS_ProcessCommandPacket(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_VerifyCmdLength(), CFE_SB_ProcessCmdPipePkt(), CFE_SB_SendMsgFull(), and CFE_SB_VerifyCmdLength().

13.60.3.8 CFE_SB_GetMsgTime()

```
CFE_TIME_SysTime_t CFE_SB_GetMsgTime (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine gets the time from a software bus message.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will return a zero time.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The system time included in the software bus message header (if present), otherwise, returns a time value of zero.

Returns

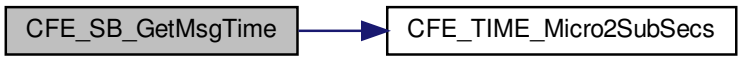
See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 290 of file cfe_sb_util.c.

References CCSDS_CMD, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CFE_TIME_Micro2SubSecs(), CFE_SB_Msg_t::Hdr, CCSDS_TelemetryPacket_t::Sec, CFE_TIME_SysTime_t::Seconds, CFE_TIME_SysTime_t::Subseconds, and CCSDS_TlmSecHdr_t::Time.

Here is the call graph for this function:



13.60.3.9 CFE_SB_GetPipeIdByName()

```
int32 CFE_SB_GetPipeIdByName (
            CFE_SB_PipeId_t * PipeIdPtr,
            const char * PipeName )
```

Description

This routine finds the pipe id for a pipe name.

Parameters

in	*PipeName*	The name of the pipe.
out	*PipeIdPtr*	The PipeId for that name.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_INVALID_PIPE	

Returns

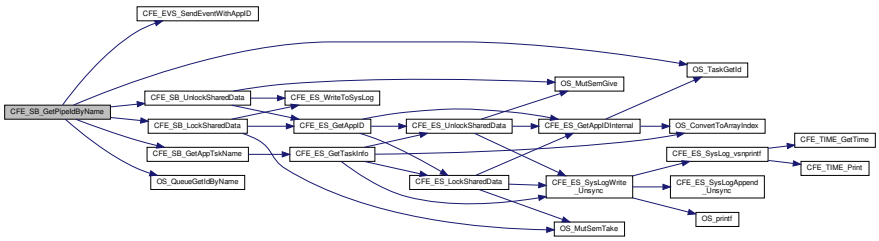
See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_PIPEOPTS_IGNOREMINE

Definition at line 628 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GETPIPEIDBYNAME_EID, CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID, CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::GetPipeIdByNameErrorCounter, cfe_sb_t::HKTlmMsg, CFE_SB_PipeD_t::InUse, NULL, OS_MAX_API_NAME, OS_QueueGetIdByName(), OS_SUCCESS, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_PipeD_t::PipeId, cfe_sb_t::PipeTbl, and CFE_SB_PipeD_t::SysQueueId.

Here is the call graph for this function:



13.60.3.10 CFE_SB_GetPipeName()

```
int32 CFE_SB_GetPipeName (
            char * PipeNameBuf,
            size_t PipeNameSize,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine finds the pipe name for a pipe id.

Parameters

out	*PipeNameBuf*	The buffer to receive the pipe name.
in	*PipeNameSize*	The size (in chars) of the PipeName buffer.
in	*PipeId*	The PipeId for that name.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_INVALID_PIPE	

Returns

See also

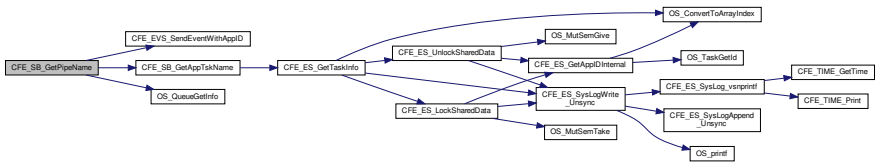
CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

Definition at line 570 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GETPIPENAME_EID, CFE_SB_GETPIPENAME_ID_ERR_EID, CFE_SB_GETPIPENAME_NULL_PTR_EID, CFE_SUCCESS, OS_queue_prop_t::name, NULL, OS_MAX_API_NAME, OS_QueueGetInfo(), OS_SUCCESS, cfe_sb_t::PipeTbl, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SendRtgInfo(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.60.3.11 CFE_SB_GetPipeOpts()

```
int32 CFE_SB_GetPipeOpts (
            CFE_SB_PipeId_t PipeId,
            uint8 * OptPtr )
```

Description

This routine gets the current options on a pipe.

Parameters

in	*PipeId*	The pipe ID of the pipe to get options from.
out	∗*OptPtr*	A bit field of options.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.

Returns

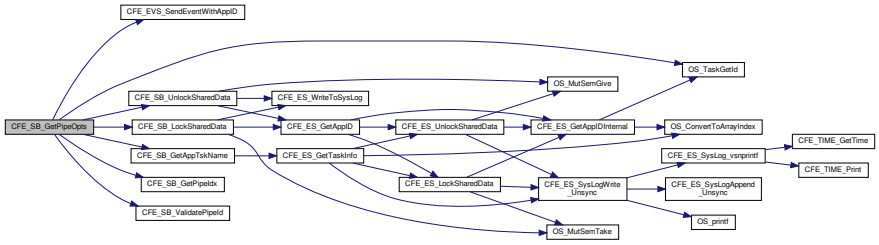
See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName CFE_SB_PIP↩ EOPTS_IGNOREMINE

Definition at line 509 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_Send↩ EventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), C↩ FE_SB_GETPIPEOPTS_EID, CFE_SB_GETPIPEOPTS_ID_ERR_EID, CFE_SB_GETPIPEOPTS_PTR_ERR_EID, CFE_SB_INVALID_PIPE, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, NULL, CFE_SB_PipeD_t::Opts, OS_MAX_API_NAME, OS_TaskGetId(), CF↩ E_SB_HousekeepingTlm_t::Payload, CFE_SB_HousekeepingTlm_Payload_t::PipeOptsErrorCounter, and cfe_sb_t::↩ PipeTbl.

Here is the call graph for this function:



13.60.3.12 CFE_SB_GetTotalMsgLength()

```
uint16 CFE_SB_GetTotalMsgLength (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the total size of the software bus message.

Assumptions, External Events, and Notes:

- For the CCSDS implementation of this API, the size is derived from the message header.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The total size (in bytes) of the software bus message, including headers.

Returns

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 242 of file cfe_sb_util.c.

References CCSDS_RD_LEN, and CFE_SB_Msg_t::Hdr.

Referenced by CFE_ES_VerifyCmdLength(), CFE_EVS_VerifyCmdLength(), CFE_SB_GetUserDataLength(), CFE_SB_SendMsgFull(), and CFE_SB_VerifyCmdLength().

13.60.3.13 CFE_SB_GetUserData()

```
void* CFE_SB_GetUserData (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine returns a pointer to the user data portion of a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function and avoid hard coding offsets into their SB message buffers.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message.

A pointer to the first byte of user data within the software bus message.

Returns

See also

CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 154 of file cfe_sb_util.c.

References CFE_SB_MsgHdrSize().

Here is the call graph for this function:

13.60.3.14 CFE_SB_GetUserDataLength()

```
uint16 CFE_SB_GetUserDataLength (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the size of the user data in a software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The size (in bytes) of the user data in the software bus message.

Returns

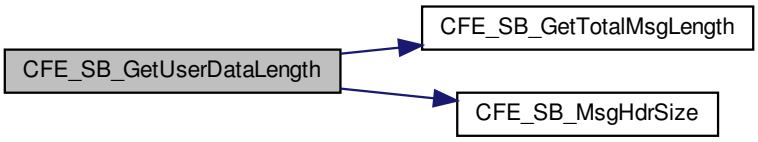
See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_SetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 183 of file cfe_sb_util.c.

References CFE_SB_GetTotalMsgLength(), and CFE_SB_MsgHdrSize().

Here is the call graph for this function:



13.60.3.15 CFE_SB_InitMsg()

```
void CFE_SB_InitMsg (
            void * MsgPtr,
            CFE_SB_MsgId_t MsgId,
            uint16 Length,
            bool Clear )
```

Description

This routine fills in the header information needed to create a valid software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that will contain the message. This will point to the first byte of the message header. The `void*` data type allows the calling routine to use any data type when declaring its message buffer.
in	*MsgId*	The message ID to put in the message header.
in	*Length*	The total number of bytes of message data, including the SB message header .
in	*Clear*	A flag indicating whether to clear the rest of the message: • true - fill sequence count and packet data with zeroes. • false - leave sequence count and packet data unchanged.

See also

CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode

Definition at line 60 of file cfe_sb_util.c.

References CCSDS_CLR_PRI_HDR, CCSDS_INIT_SEQFLG, CCSDS_RD_SEQ, CCSDS_WR_LEN, CCSDS_WR_SEQ, CCSDS_WR_SEQFLG, CCSDS_WR_SHDR, and CFE_SB_SetMsgId().

Referenced by CFE_ES_TaskInit(), CFE_EVS_EarlyInit(), CFE_SB_AppInit(), CFE_SB_EarlyInit(), and EVS_GenerateEventTelemetry().

Here is the call graph for this function:

13.60.3.16 CFE_SB_MessageStringGet()

```
int32 CFE_SB_MessageStringGet (
            char * DestStringPtr,
            const char * SourceStringPtr,
            const char * DefaultString,
            uint32 DestMaxSize,
            uint32 SourceMaxSize )
```

Definition at line 612 of file cfe_sb_util.c.

References CFE_SB_BAD_ARGUMENT, and NULL.

Referenced by CFE_ES_AppCreate(), CFE_ES_DeleteCDSCmd(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_RestartAppCmd(), CFE_ES_ShellCmd(), CFE_ES_StartAppCmd(), CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_WriteERLogCmd(), CFE_ES_WriteSyslogCmd(), CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

13.60.3.17 CFE_SB_MessageStringSet()

```
int32 CFE_SB_MessageStringSet (
            char * DestStringPtr,
            const char * SourceStringPtr,
            uint32 DestMaxSize,
            uint32 SourceMaxSize )
```

Definition at line 673 of file cfe_sb_util.c.

13.60.3.18 CFE_SB_MsgId_Equal()

```
static bool CFE_SB_MsgId_Equal (
            CFE_SB_MsgId_t MsgId1,
            CFE_SB_MsgId_t MsgId2 )  [inline], [static]
```

Description

In cases where the CFE_SB_MsgId_t type is not a simple integer type, it may not be possible to do a direct equality check. This inline function provides an abstraction for the equality check between two CFE_SB_MsgId_t values.

Applications should transition to using this function to compare MsgId values for equality to remain compatible with future versions of cFE.

Returns

true if equality checks passed, false otherwise.

Definition at line 1302 of file cfe_sb.h.

13.60.3.19 CFE_SB_MsgIdToValue()

```
static CFE_SB_MsgId_Atom_t CFE_SB_MsgIdToValue (
            CFE_SB_MsgId_t MsgId )  [inline], [static]
```

Description

In cases where the CFE_SB_MsgId_t type is not a simple integer type, it is not possible to directly display the value in a printf-style statement, use it in a switch() statement, or other similar use cases.

This inline function provides the ability to map a CFE_SB_MsgId_t type back into a simple integer value.

Applications should transition to using this function wherever a CFE_SB_MsgId_t type needs to be used as an integer.

Assumptions and Notes:

This negates the type safety that was gained by using a non- integer type for the CFE_SB_MsgId_t value. This should only be used in specific cases such as UI display (printf, events, etc) where the value is being sent externally. Any internal API calls should be updated to use the CFE_SB_MsgId_t type directly, rather than an integer type.

Returns

Integer representation of the CFE_SB_MsgId_t

Definition at line 1333 of file cfe_sb.h.

13.60.3.20 CFE_SB_PassMsg()

```
int32 CFE_SB_PassMsg (
            CFE_SB_Msg_t * MsgPtr )
```

Description

This routine sends the specified message to all subscribers. The software bus will read the message ID from the message header to determine which pipes should receive the message. This routine is intended to pass messages not generated by the sending application.

Assumptions, External Events, and Notes:

- This routine will not normally wait for the receiver tasks to process the message before returning control to the caller's task.
- However, if a higher priority task is pending and subscribed to this message, that task may get to run before CFE_SB_PassMsg returns control to the caller.
- Unlike CFE_SB_SendMsg this routine will preserve the source sequence counter in a telemetry message.

Parameters

in	*MsgPtr*	A pointer to the message to be sent. This must point to the first byte of the software bus message header (CFE_SB_Msg_t).

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF↩ _MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

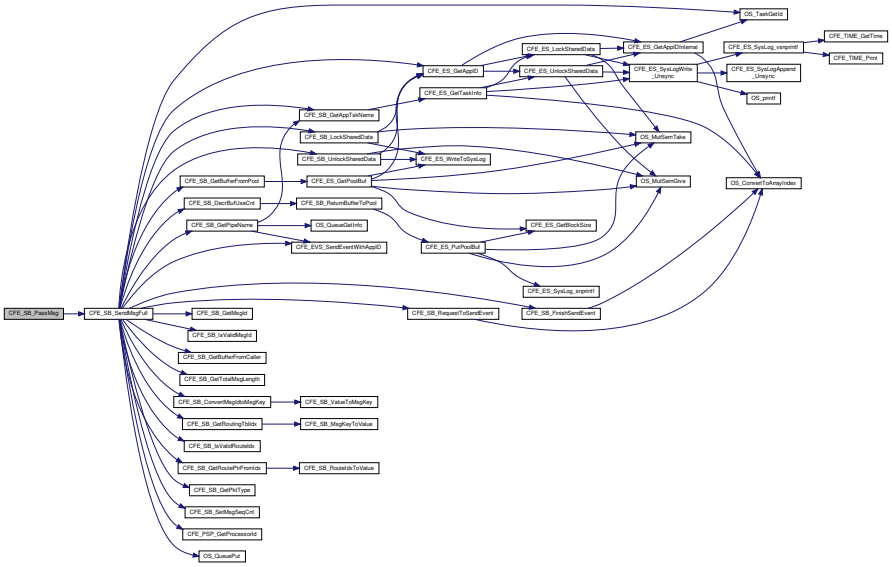
See also

CFE_SB_RcvMsg, CFE_SB_ZeroCopySend, CFE_SB_SendMsg

Definition at line 1353 of file cfe_sb_api.c.

References CFE_SB_DO_NOT_INCREMENT, CFE_SB_SEND_ONECOPY, and CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.60.3.21 CFE_SB_RcvMsg()

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

Description

This routine retrieves the next message from the specified pipe. If the pipe is empty, this routine will block until either a new message comes in or the timeout value is reached.

Assumptions, External Events, and Notes:

Note - If an error occurs in this API, the ∗BufPtr value may be NULL or random. Therefore, it is recommended that the return code be tested for CFE_SUCCESS before processing the message.

Parameters

in	*BufPtr*	A pointer to a local variable of type CFE_SB_MsgPtr_t. Typically a caller declares a ptr of type CFE_SB_Msg_t (i.e. CFE_SB_Msg_t ∗Ptr) then gives the address of that pointer (&Ptr) as this parmeter. After a successful receipt of a message, ∗BufPtr will point to the first byte of the software bus message header. This should be used as a read-only pointer (in systems with an MMU, writes to this pointer may cause a memory protection fault). The ∗BufPtr is valid only until the next call to CFE_SB_RcvMsg for the same pipe.
in	*PipeId*	The pipe ID of the pipe containing the message to be obtained.
in	*TimeOut*	The number of milliseconds to wait for a new message if the pipe is empty at the time of the call. This can also be set to CFE_SB_POLL for a non-blocking receive or CFE_SB_PEND_FOREVER to wait forever for a message to arrive.
out	∗*BufPtr*	A pointer to the message obtained from the pipe. Valid only until the next call to CFE_SB_RcvMsg for the same pipe.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_TIME_OUT	In CFE_SB_RcvMsg, this return value indicates that a packet has not been received in the time given in the "timeout" parameter.
CFE_SB_PIPE_RD_ERR	This return value indicates an error at the Queue read level. This error typically cannot be corrected by the caller. Some possible causes are: queue was not properly initialized or created, the number of bytes read from the queue was not the number of bytes requested in the read. The queue id is invalid. Similar errors regarding the pipe will be caught by higher level code in the Software Bus.
CFE_SB_NO_MESSAGE	When "Polling" a pipe for a message in CFE_SB_RcvMsg, this return value indicates that there was not a message on the pipe.

Returns

See also

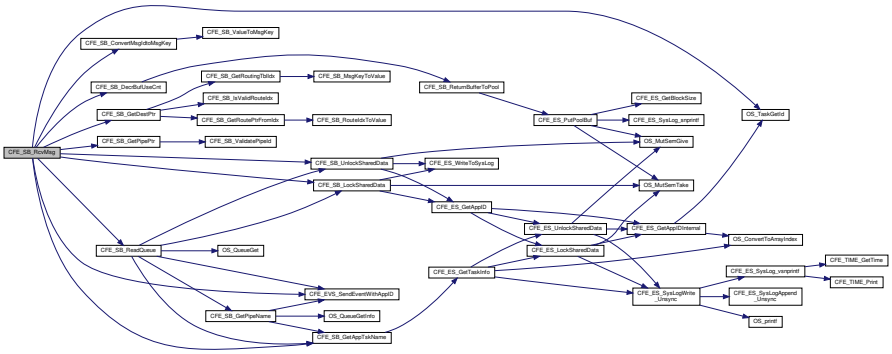
CFE_SB_SendMsg, CFE_SB_ZeroCopySend

Definition at line 1739 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_SB_DestinationD_t::BuffCount, CFE_SB_BufferD_t::Buffer, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_BAD_PIPEID_EID, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_DecrBufUseCnt(), CFE_SB_GetAppTskName(), CFE_SB_GetDestPtr(), CFE_SB_GetPipePtr(), CFE_SB_LockSharedData(), CFE_SB_RCV_BAD_ARG_EID, CFE_SB_ReadQueue(), CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_PipeD_t::CurrentBuff, cfe_sb_t::HKTlmMsg, CFE_SB_PipeDepthStats_t::InUse, CFE_SB_BufferD_t::MsgId, CFE_SB_HousekeepingTlm_Payload_t::MsgReceiveErrorCounter, NULL, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_PipeD_t::PipeId, cfe_sb_t::StatTlmMsg, and CFE_SB_PipeD_t::ToTrashBuff.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), CFE_SB_DeletePipeFull(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.60.3.22 CFE_SB_SendMsg()

```
int32 CFE_SB_SendMsg (
            CFE_SB_Msg_t * MsgPtr )
```

Description

This routine sends the specified message to all subscribers. The software bus will read the message ID from the message header to determine which pipes should receive the message.

Assumptions, External Events, and Notes:

- This routine will not normally wait for the receiver tasks to process the message before returning control to the caller's task.
- However, if a higher priority task is pending and subscribed to this message, that task may get to run before CFE_SB_SendMsg returns control to the caller.
- This function tracks and increments the source sequence counter of a telemetry message.

Parameters

in	*MsgPtr*	A pointer to the message to be sent. This must point to the first byte of the software bus message header (CFE_SB_Msg_t).

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF↩ _MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

See also

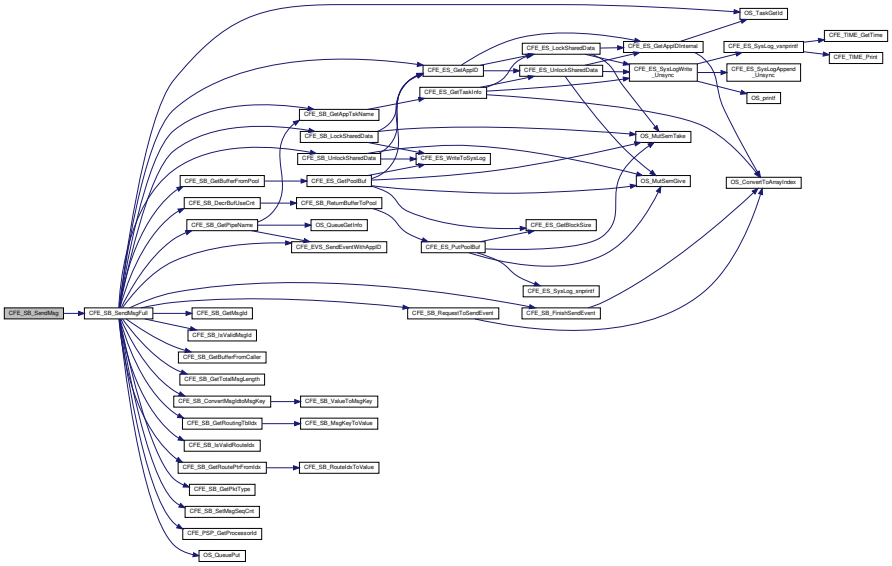
CFE_SB_RcvMsg, CFE_SB_ZeroCopySend, CFE_SB_PassMsg

Definition at line 1318 of file cfe_sb_api.c.

References CFE_SB_INCREMENT_TLM, CFE_SB_SEND_ONECOPY, and CFE_SB_SendMsgFull().

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_QueryOneCmd(), CFE_ES_SendMemPoolStatsCmd(), C↩ FE_ES_ShellOutputCommand(), CFE_EVS_ReportHousekeepingCmd(), CFE_SB_SendHKTlmCmd(), CFE_SB_↩ SendPrevSubsCmd(), CFE_SB_SendStatsCmd(), CFE_SB_SubscribeFull(), and EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.60.3.23 CFE_SB_SetCmdCode()

```
int32 CFE_SB_SetCmdCode (
            CFE_SB_MsgPtr_t MsgPtr,
            uint16 CmdCode )
```

Description

This routine sets the command code of a software bus message (if SB messages are implemented as CCSDS packets, this will be the function code).

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a command code field, then this routine will do nothing to the message contents and will return CFE_SB_WRONG_MSG_TYPE.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*CmdCode*	The command code to include in the message.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_WRONG_MSG_TYPE	This error code will be returned when a request such as ...SetMsgTime is made on a packet that does not include a field for msg time.

Returns

See also

CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_GetCmdCode, CFE_SB_InitMsg

Definition at line 476 of file cfe_sb_util.c.

References CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CCSDS_WR_FC, CFE_SB_WRONG_MSG_TYPE, CFE_SUCCESS, CFE_SB_Msg_t::Hdr, and CCSDS_CommandPacket_t::Sec.

13.60.3.24 CFE_SB_SetMsgId()

```
void CFE_SB_SetMsgId (
            CFE_SB_MsgPtr_t MsgPtr,
            CFE_SB_MsgId_t MsgId )
```

Description

This routine sets the Message ID in a software bus message header.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*MsgId*	The message ID to put into the message header.

The software bus Message ID from the message header.

Returns

See also

CFE_SB_GetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 183 of file cfe_sb_msg_id_util.c.

References CCSDS_CLR_SEC_APIDQ, CCSDS_WR_APID, CCSDS_WR_EDS_VER, CCSDS_WR_ENDIAN, CCSDS_WR_PLAYBACK, CCSDS_WR_SID, CCSDS_WR_SUBSYSTEM_ID, CCSDS_WR_SYSTEM_ID, CCSDS_WR_TYPE, CCSDS_WR_VERS, CFE_PLATFORM_ENDIAN, CFE_SB_RD_APID_FROM_MSGID, CFE_SB_RD_SUBSYS_ID_FROM_MSGID, CFE_SB_RD_TYPE_FROM_MSGID, CFE_SPACECRAFT_ID, CFE_SB_Msg_t::Hdr, CCSDS_SpacePacket_t::Hdr, and CFE_SB_Msg_t::SpacePacket.

Referenced by CFE_SB_InitMsg().

13.60.3.25 CFE_SB_SetMsgTime()

```
int32 CFE_SB_SetMsgTime (
            CFE_SB_MsgPtr_t MsgPtr,
            CFE_TIME_SysTime_t Time )
```

Description

This routine sets the time of a software bus message. Most applications will want to use CFE_SB_TimeStampMsg instead of this function. But, when needed, CFE_SB_SetMsgTime can be used to send a group of SB messages with identical time stamps.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will do nothing to the message contents and will return CFE_SB_WRONG_MSG_TYPE.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*Time*	The time to include in the message. This will usually be a time returned by the function CFE_TIME_GetTime().

CFE_SUCCESS	Operation was performed successfully
CFE_SB_WRONG_MSG_TYPE	This error code will be returned when a request such as ...SetMsgTime is made on a packet that does not include a field for msg time.

Returns

See also

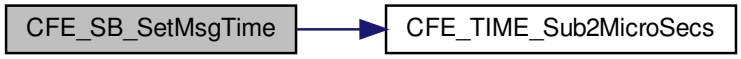
CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 358 of file cfe_sb_util.c.

References CCSDS_CMD, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CFE_SB_WRONG_MSG_TYPE, CFE_SUCCESS, CFE_TIME_Sub2MicroSecs(), CFE_SB_Msg_t::Hdr, CCSDS_TelemetryPacket_t::Sec, CFE_TIME_SysTime_t::Seconds, CFE_TIME_SysTime_t::Subseconds, and CCSDS_TlmSecHdr_t::Time.

Referenced by CFE_SB_TimeStampMsg(), and EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.60.3.26 CFE_SB_SetPipeOpts()

```
int32 CFE_SB_SetPipeOpts (
            CFE_SB_PipeId_t PipeId,
            uint8 Opts )
```

Description

This routine sets (or clears) options to alter the pipe's behavior. Options are (re)set every call to this routine.

Parameters

in	*PipeId*	The pipe ID of the pipe to set options on.
in	*Opts*	A bit field of options.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.

Returns

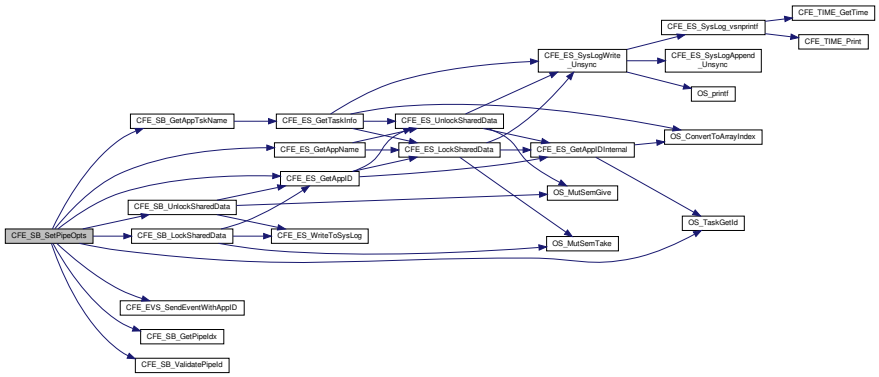
See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_GetPipeOpts CFE_SB_GetPipeIdByName CFE_SB_PIP↩ EOPTS_IGNOREMINE

Definition at line 433 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_ES_GetAppName(), CFE_EVS_↩ EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_↩ ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), CFE_SB_INVALID_PIPE, CFE_SB_LockShared↩ Data(), CFE_SB_SETPIPEOPTS_EID, CFE_SB_SETPIPEOPTS_ID_ERR_EID, CFE_SB_SETPIPEOPTS_OWNE↩ R_ERR_EID, CFE_SB_UnlockSharedData(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, C↩ FE_SB_PipeD_t::Opts, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_↩ HousekeepingTlm_Payload_t::PipeOptsErrorCounter, and cfe_sb_t::PipeTbl.

Here is the call graph for this function:



13.60.3.27 CFE_SB_SetTotalMsgLength()

```
void CFE_SB_SetTotalMsgLength (
            CFE_SB_MsgPtr_t MsgPtr,
            uint16 TotalLength )
```

Description

This routine sets the field in the SB message header that determines the total length of the message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function rather than trying to poke a length value directly into their SB message buffers.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*TotalLength*	The length to set (total size of the message, in bytes, including headers).

See also

CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 267 of file cfe_sb_util.c.

References CCSDS_WR_LEN, and CFE_SB_Msg_t::Hdr.

13.60.3.28 CFE_SB_SetUserDataLength()

```
void CFE_SB_SetUserDataLength (
            CFE_SB_MsgPtr_t MsgPtr,
            uint16 DataLength )
```

Description

This routine sets the field in the SB message header that determines the size of the user data in a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function rather than trying to poke a length value directly into their SB message buffers.

Assumptions, External Events, and Notes:

- You must set a valid message ID in the SB message header before calling this function.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*DataLength*	The length to set (size of the user data, in bytes).

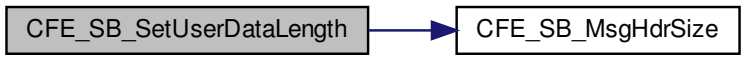
See also

CFE_SB_SetMsgId, CFE_SB_GetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 214 of file cfe_sb_util.c.

References CCSDS_WR_LEN, CFE_SB_MsgHdrSize(), and CFE_SB_Msg_t::Hdr.

Here is the call graph for this function:



13.60.3.29 CFE_SB_Subscribe()

```
int32 CFE_SB_Subscribe (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine adds the specified pipe to the destination list for the specified message ID. This is the same as CFE_SB_SubscribeEx with the Quality field set to CFE_SB_Default_Qos and MsgLim set to CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT (4).

Assumptions, External Events, and Notes:

Note: As subscriptions are received, the destinations are added to the head of a linked list. During the sending of a message, the list is traversed beginning at the head of the list. Therefore the message will first be sent to the last subscriber. If an application has timing constraints and needs to receive a message in the shortest possible time, the developer may consider holding off its subscription until other applications have subscribed to the message.

Parameters

in	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_MAX_MSGS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.

CFE_SB_MAX_DESTS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER↩_PKT.
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_S↩B_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

See also

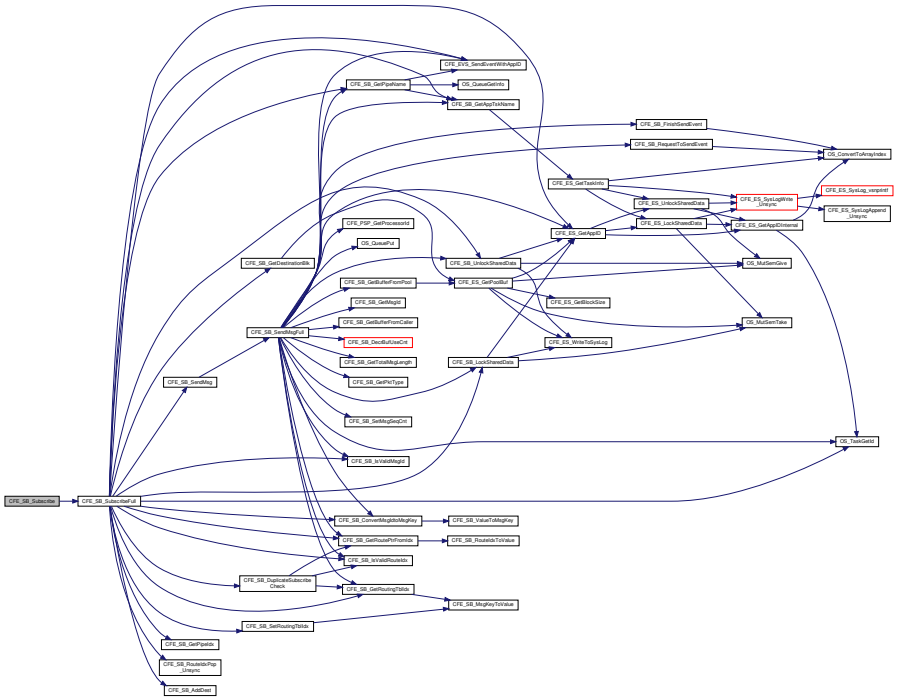
CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

Definition at line 828 of file cfe_sb_api.c.

References CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, CFE_SB_Default_Qos, CFE_SB_GLOBAL, and CFE_S↩B_SubscribeFull().

Referenced by CFE_SB_AppInit().

Here is the call graph for this function:



13.60.3.30 CFE_SB_SubscribeEx()

```
int32 CFE_SB_SubscribeEx (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            CFE_SB_Qos_t Quality,
            uint16 MsgLim )
```

Description

This routine adds the specified pipe to the destination list associated with the specified message ID.

Assumptions, External Events, and Notes:

Note: As subscriptions are received, the destinations are added to the head of a linked list. During the sending of a message, the list is traversed beginning at the head of the list. Therefore the message will first be sent to the last subscriber. If an application has timing constraints and needs to receive a message in the shortest possible time, the developer may consider holding off its subscription until other applications have subscribed to the message.

Parameters

in	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.
in	*Quality*	The requested Quality of Service (QoS) required of the messages. Most callers will use CFE_SB_Default_Qos for this parameter.
in	*MsgLim*	The maximum number of messages with this Message ID to allow in this pipe at the same time.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_MAX_MSGS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.
CFE_SB_MAX_DESTS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER_PKT.
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

See also

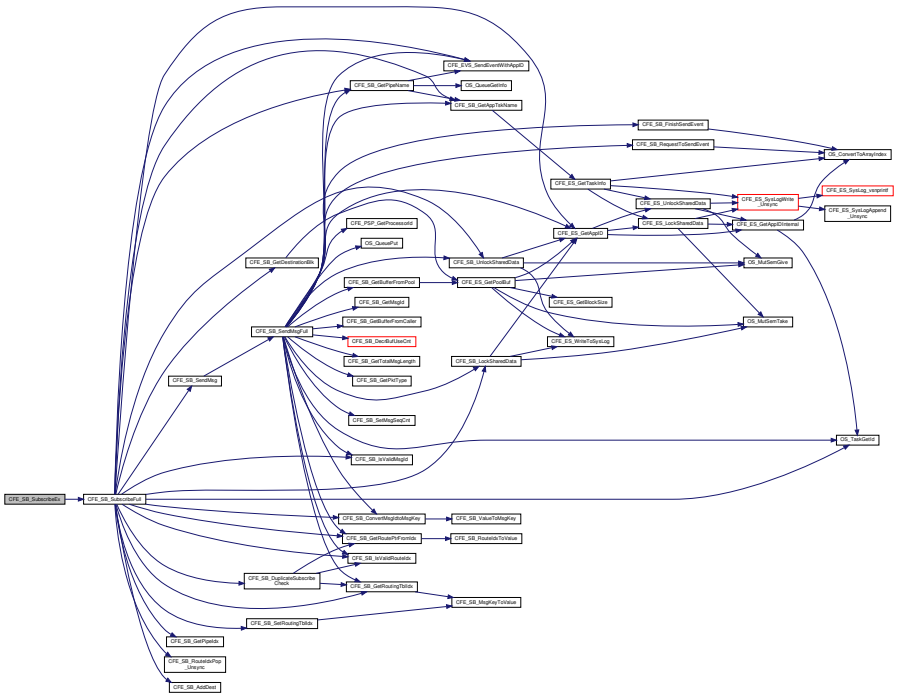
CFE_SB_Subscribe, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

Definition at line 750 of file cfe_sb_api.c.

References CFE_SB_GLOBAL, and CFE_SB_SubscribeFull().

Referenced by CFE_ES_TaskInit(), and CFE_EVS_TaskInit().

Here is the call graph for this function:



13.60.3.31 CFE_SB_SubscribeLocal()

```
int32 CFE_SB_SubscribeLocal (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            uint16 MsgLim )
```

Description

This routine adds the specified pipe to the destination list for the specified message ID. This is similar to CFE_SB_SubscribeEx with the Quality field set to CFE_SB_Default_Qos and MsgLim set to CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, but will not report the subscription. Subscription Reporting is enabled for interprocessor communication by way of the Software Bus Network (SBN) Application.

Assumptions, External Events, and Notes:

- This API is typically only used by Software Bus Network (SBN) Application

Parameters

in	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.
in	*MsgLim*	The maximum number of messages with this Message ID to allow in this pipe at the same time.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_MAX_MSGS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.
CFE_SB_MAX_DESTS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER_PKT.
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

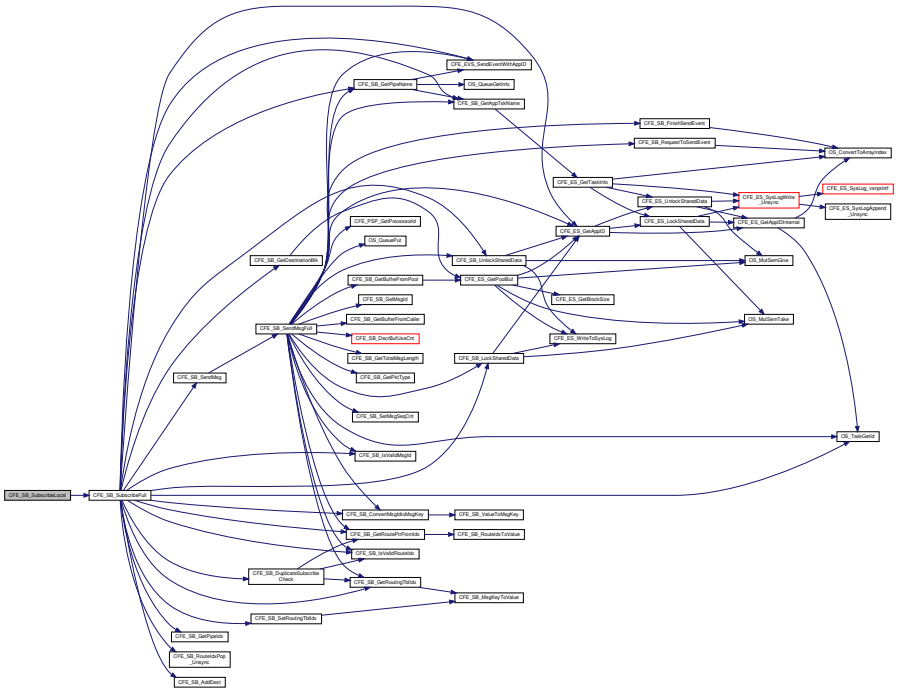
See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

Definition at line 790 of file cfe_sb_api.c.

References CFE_SB_Default_Qos, CFE_SB_LOCAL, and CFE_SB_SubscribeFull().

Here is the call graph for this function:



13.60.3.32 CFE_SB_TimeStampMsg()

```
void CFE_SB_TimeStampMsg (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine sets the time of a software bus message with the current spacecraft time. This will be the same time that is returned by the function CFE_TIME_GetTime.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will do nothing.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

See also

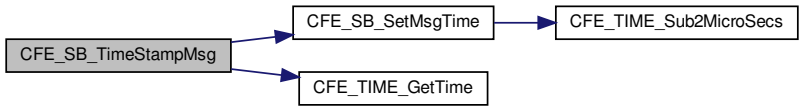
CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 424 of file cfe_sb_util.c.

References CFE_SB_SetMsgTime(), and CFE_TIME_GetTime().

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_QueryOneCmd(), CFE_ES_SendMemPoolStatsCmd(), CFE_ES_ShellOutputCommand(), CFE_EVS_ReportHousekeepingCmd(), CFE_SB_SendHKTlmCmd(), and CFE_SB_SendStatsCmd().

Here is the call graph for this function:



13.60.3.33 CFE_SB_Unsubscribe()

```
int32 CFE_SB_Unsubscribe (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine removes the specified pipe from the destination list for the specified message ID.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgId*	The message ID of the message to be unsubscribed.
in	*PipeId*	The pipe ID of the pipe the subscribed message should no longer be sent to.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_NO_SUBSCRIBERS	This error code is returned by the CFE_SB_Unsubscribe API if there has not been an entry in the routing tables for the MsgId/PipeId given as parameters.
CFE_SB_INTERNAL_ERR	This error code will be returned by the CFE_SB_Subscribe API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Returns

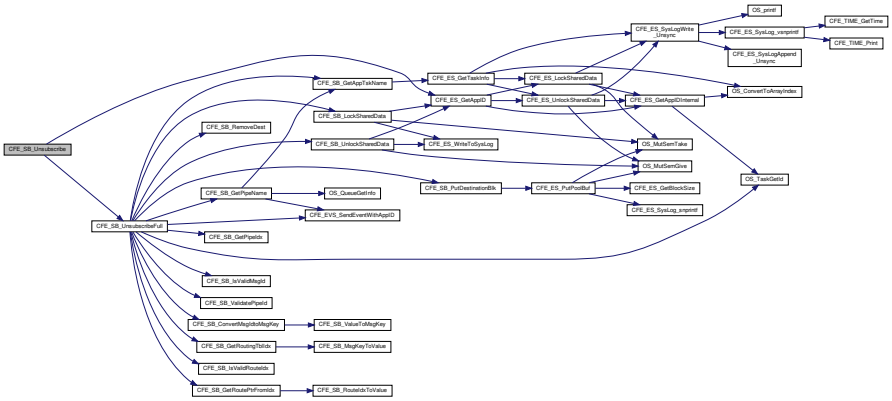
See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_UnsubscribeLocal

Definition at line 1075 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), CFE_SB_GLOBAL, and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.60.3.34 CFE_SB_UnsubscribeLocal()

```
int32 CFE_SB_UnsubscribeLocal (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine removes the specified pipe from the destination list for the specified message ID on the current CPU.

Assumptions, External Events, and Notes:

- This API is typically only used by Software Bus Network (SBN) Application

Parameters

in	*MsgId*	The message ID of the message to be unsubscribed.
in	*PipeId*	The pipe ID of the pipe the subscribed message should no longer be sent to.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_NO_SUBSCRIBERS	This error code is returned by the CFE_SB_Unsubscribe API if there has not been an entry in the routing tables for the MsgId/PipeId given as parameters.
CFE_SB_INTERNAL_ERR	This error code will be returned by the CFE_SB_Subscribe API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Returns

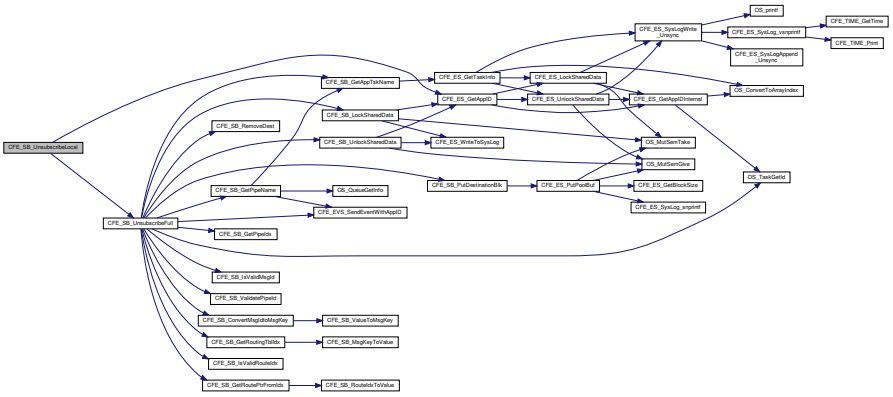
See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe

Definition at line 1110 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), CFE_SB_LOCAL, and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.60.3.35 CFE_SB_ValidateChecksum()

```
bool CFE_SB_ValidateChecksum (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine calculates the expected checksum of a software bus message according to an implementation-defined algorithm. Then, it checks the calculated value against the value in the message's checksum. If the checksums do not match, this routine will generate an event message reporting the error.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a checksum field, then this routine will always return `true`.

Parameters

`in`	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

true	The checksum field in the packet is valid.
false	The checksum field in the packet is not valid or the message type is wrong.

Returns

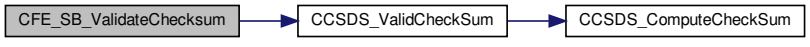
See also

CFE_SB_GenerateChecksum, CFE_SB_GetChecksum

Definition at line 581 of file cfe_sb_util.c.

References CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CCSDS_ValidCheckSum(), and CFE_SB_Msg↩_t::Hdr.

Here is the call graph for this function:



13.60.3.36 CFE_SB_ValueToMsgId()

```
static CFE_SB_MsgId_t CFE_SB_ValueToMsgId (
            CFE_SB_MsgId_Atom_t MsgIdValue )  [inline], [static]
```

Description:

In cases where the CFE_SB_MsgId_t type is not a simple integer type, it is not possible to directly use an integer value supplied via a define or similar method.

This inline function provides the ability to map an integer value into a corresponding CFE_SB_MsgId_t value.

Applications should transition to using this function wherever an integer needs to be used for a CFE_SB_MsgId_t.

Assumptions and Notes:

This negates the type safety that was gained by using a non- integer type for the CFE_SB_MsgId_t value. This should only be used in specific cases where the value is coming from an external source. Any internal API calls should be updated to return the CFE_SB_MsgId_t type directly, rather than an integer type.

Returns

CFE_SB_MsgId_t representation of the integer

Definition at line 1362 of file cfe_sb.h.

13.60.3.37 CFE_SB_ZeroCopyGetPtr()

```
CFE_SB_Msg_t* CFE_SB_ZeroCopyGetPtr (
            uint16 MsgSize,
            CFE_SB_ZeroCopyHandle_t * BufferHandle )
```

Description

This routine can be used to get a pointer to one of the software bus' internal memory buffers that are used for sending messages. The caller can use this memory buffer to build an SB message, then send it using the CFE_SB_ZeroCopySend function. This interface is more complicated than the normal CFE_SB_ZeroCopySend interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic.

Assumptions, External Events, and Notes:

1. The pointer returned by CFE_SB_ZeroCopyGetPtr is only good for one call to CFE_SB_ZeroCopySend.
2. Applications should be written as if CFE_SB_ZeroCopyGetPtr is equivalent to a `malloc()` and CFE_SB_ZeroCopySend is equivalent to a `free()`.
3. Applications must not de-reference the message pointer (for reading or writing) after the call to CFE_SB_ZeroCopySend.

Parameters

in	*MsgSize*	The size of the SB message buffer the caller wants (including the SB message header).
out	*BufferHandle*	A handle that must be supplied when sending or releasing in zero copy mode.

A pointer to a memory buffer that can be used to build one SB message for use with CFE_SB_ZeroCopySend.

Returns

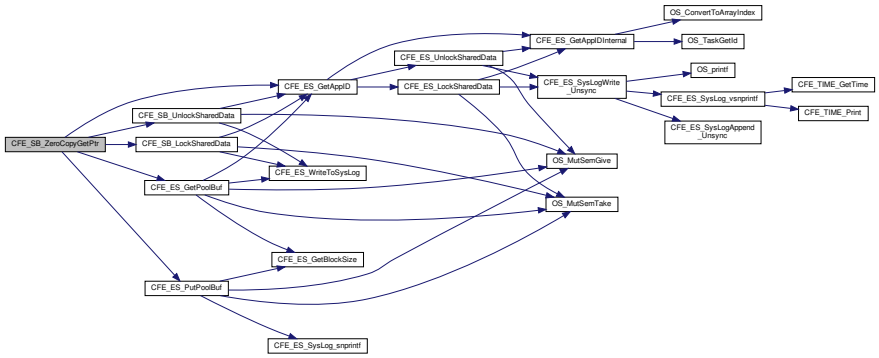
See also

CFE_SB_ZeroCopyReleasePtr, CFE_SB_ZeroCopySend

Definition at line 1962 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), CFE_ES_GetPoolBuf(), CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, CFE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, CFE_SB_StatsTlm_Payload_t::PeakSBBuffersInUse, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, cfe_sb_t::StatTlmMsg, and cfe_sb_t::ZeroCopyTail.

Here is the call graph for this function:



13.60.3.38 CFE_SB_ZeroCopyPass()

```
int32 CFE_SB_ZeroCopyPass (
            CFE_SB_Msg_t * MsgPtr,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Description

This routine sends a message that has been created directly in an internal SB message buffer by an application (after a call to CFE_SB_ZeroCopyGetPtr). This interface is more complicated than the normal CFE_SB_SendMsg interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic. This version is intended to pass messages not generated by the caller (to preserve the source sequence count).

Assumptions, External Events, and Notes:

1. The pointer returned by CFE_SB_ZeroCopyGetPtr is only good for one call to CFE_SB_ZeroCopySend or CFE_SB_ZeroCopyPass.
2. Callers must not use the same SB message buffer for multiple sends.
3. Applications should be written as if CFE_SB_ZeroCopyGetPtr is equivalent to a `malloc()` and CFE_SB_ZeroCopyPass is equivalent to a `free()`.
4. Applications must not de-reference the message pointer (for reading or writing) after the call to CFE_SB_ZeroCopyPass.
5. Unlike CFE_SB_ZeroCopySend this routine will preserve the source sequence counter in a telemetry message.

Parameters

`in`	*MsgPtr*	A pointer to the SB message to be sent.
`in`	*BufferHandle*	The handle supplied with the CFE_SB_ZeroCopyGetPtr call.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.
CFE_SB_BUFFER_INVALID	This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Returns

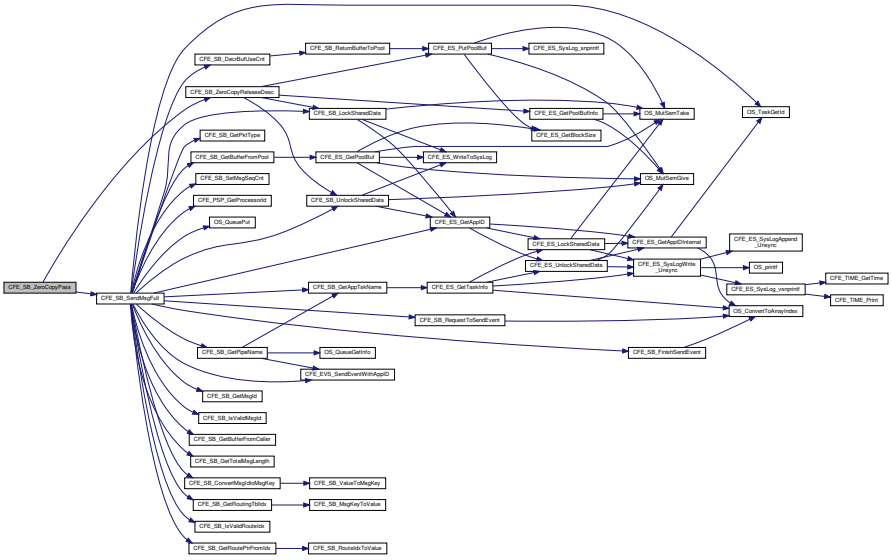
See also

CFE_SB_PassMsg, CFE_SB_ZeroCopySend, CFE_SB_ZeroCopyReleasePtr, CFE_SB_ZeroCopyGetPtr

Definition at line 2228 of file cfe_sb_api.c.

References CFE_SB_DO_NOT_INCREMENT, CFE_SB_SEND_ZEROCOPY, CFE_SB_SendMsgFull(), CFE_SB_↩ ZeroCopyReleaseDesc(), and CFE_SUCCESS.

Here is the call graph for this function:



13.60.3.39 CFE_SB_ZeroCopyReleasePtr()

```
int32 CFE_SB_ZeroCopyReleasePtr (
            CFE_SB_Msg_t * Ptr2Release,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Description

This routine can be used to release a pointer to one of the software bus' internal memory buffers.

Assumptions, External Events, and Notes:

1. This function is not needed for normal "zero copy" transfers. It is needed only for cleanup when an application gets a pointer using CFE_SB_ZeroCopyGetPtr, but (due to some error condition) never uses that pointer for a CFE_SB_ZeroCopySend

Parameters

in	*Ptr2Release*	A pointer to the SB internal buffer. This must be a pointer returned by a call to CFE_SB_ZeroCopyGetPtr, but never used in a call to CFE_SB_ZeroCopySend.
in	*BufferHandle*	This must be the handle supplied with the pointer when CFE_SB_ZeroCopyGetPtr was called.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BUFFER_INVALID	This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Returns

See also

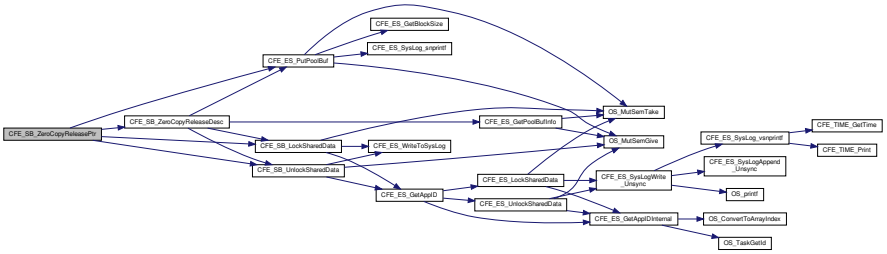
CFE_SB_ZeroCopyGetPtr, CFE_SB_ZeroCopySend

Definition at line 2074 of file cfe_sb_api.c.

References CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SB_ZeroCopyReleaseDesc(), CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_ZeroCopyReleaseAppId().

Here is the call graph for this function:



13.60.3.40 CFE_SB_ZeroCopySend()

```
int32 CFE_SB_ZeroCopySend (
            CFE_SB_Msg_t * MsgPtr,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Description

This routine sends a message that has been created directly in an internal SB message buffer by an application (after a call to CFE_SB_ZeroCopyGetPtr). This interface is more complicated than the normal CFE_SB_SendMsg interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic.

Assumptions, External Events, and Notes:

1. The pointer returned by CFE_SB_ZeroCopyGetPtr is only good for one call to CFE_SB_ZeroCopySend.
2. Callers must not use the same SB message buffer for multiple sends.
3. Applications should be written as if CFE_SB_ZeroCopyGetPtr is equivalent to a `malloc()` and CFE_SB_ZeroCopySend is equivalent to a `free()`.
4. Applications must not de-reference the message pointer (for reading or writing) after the call to CFE_SB_ZeroCopySend.
5. This function tracks and increments the source sequence counter of a telemetry message.

Parameters

`in`	*MsgPtr*	A pointer to the SB message to be sent.
`in`	*BufferHandle*	The handle supplied with the CFE_SB_ZeroCopyGetPtr call.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.
CFE_SB_BUFFER_INVALID	This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Returns

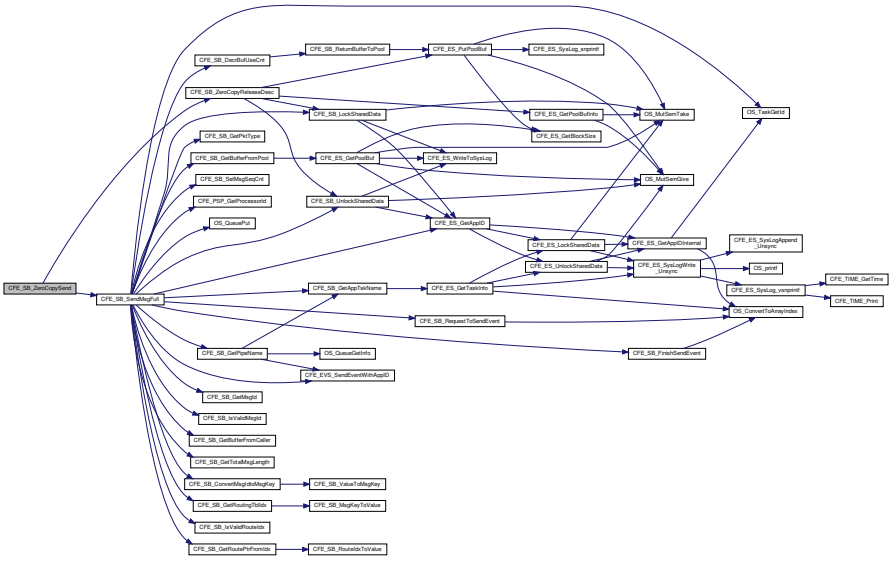
See also

CFE_SB_SendMsg, CFE_SB_RcvMsg, CFE_SB_ZeroCopyReleasePtr, CFE_SB_ZeroCopyGetPtr

Definition at line 2189 of file cfe_sb_api.c.

References CFE_SB_INCREMENT_TLM, CFE_SB_SEND_ZEROCOPY, CFE_SB_SendMsgFull(), CFE_SB_Zero↩CopyReleaseDesc(), and CFE_SUCCESS.

Here is the call graph for this function:



13.60.4 Variable Documentation

13.60.4.1 CFE_SB_Default_Qos

```
CFE_SB_Qos_t CFE_SB_Default_Qos
```

Message Sender Identification Type Definition

Parameter used in CFE_SB_GetLastSenderId API which allows the receiver of a message to validate the sender of the message.

Definition at line 50 of file cfe_sb_task.c.

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), CFE_SB_EarlyInit(), CFE_SB_Subscribe(), and CFE_SB_↩SubscribeLocal().

13.61 cfe/fsw/cfe-core/src/inc/cfe_sb_events.h File Reference

Macros

- #define CFE_SB_MAX_EID 67
- #define CFE_SB_INIT_EID 1

  'cFE SB Initialized'
- #define CFE_SB_CR_PIPE_BAD_ARG_EID 2

  'CreatePipeErr:Bad Input Arg:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'
- #define CFE_SB_MAX_PIPES_MET_EID 3

  'CreatePipeErr:Max Pipes(%d)In Use.app %s'
- #define CFE_SB_CR_PIPE_ERR_EID 4

  'CreatePipeErr:OS_QueueCreate returned %d,app %s'
- #define CFE_SB_PIPE_ADDED_EID 5

  'Pipe Created:name %s,id %d,app %s'
- #define CFE_SB_SETPIPEOPTS_ID_ERR_EID 55

  'SetPipeOptsErr:Invalid pipe id (%d).app %s'
- #define CFE_SB_SETPIPEOPTS_OWNER_ERR_EID 56

  'SetPipeOptsErr:Caller not owner (%d).app %s'
- #define CFE_SB_SETPIPEOPTS_EID 57

  'SetPipeOpts: Options set (%d). app %s'
- #define CFE_SB_GETPIPEOPTS_ID_ERR_EID 58

  'GetPipeOptsErr:Invalid pipe id (%d).app %s'
- #define CFE_SB_GETPIPEOPTS_PTR_ERR_EID 59

  'GetPipeOptsErr:Invalid opts ptr.app %s'
- #define CFE_SB_GETPIPEOPTS_EID 60

  'GetPipeOpts: Options retrieved. app %s'
- #define CFE_SB_GETPIPENAME_EID 62

  'GetPipeName: Name retrieved. NameOut %s,Id %d, app %s'
- #define CFE_SB_GETPIPENAME_NULL_PTR_EID 63

  'GetPipeName: Null ptr error. Id %d, app %s'
- #define CFE_SB_GETPIPENAME_ID_ERR_EID 64

  'GetPipeName: Id error. NameOut %s,Id %d, app %s'
- #define CFE_SB_GETPIPEIDBYNAME_EID 65

  'GetPipeIdByName: ID retrieved. Name %s,IdOut 0x%x, app %s'
- #define CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID 66

  'GetPipeIdByName Err:Bad input argument,Name 0x%x,IdOut 0xx,App %s'
- #define CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID 67

  'GetPipeIdByName Err:Name not found,Name %s,IdOut 0xx,App %s'
- #define CFE_SB_SUB_ARG_ERR_EID 6

  'Subscribe Err:Bad Arg,MsgId 0x%x,PipeId %d,app %s,scope %d'
- #define CFE_SB_DUP_SUBSCRIP_EID 7

  'Duplicate Subscription,MsgId 0x%x on %s pipe,app %s'
- #define CFE_SB_MAX_MSGS_MET_EID 8

  'Subscribe Err:Max Msgs(%d)In Use,MsgId 0x%x,pipe %s,app %s'
- #define CFE_SB_MAX_DESTS_MET_EID 9

  'Subscribe Err:Max Dests(%d)In Use For Msg 0x%x,pipe %s,app %s'

- #define CFE_SB_SUBSCRIPTION_RCVD_EID 10

  'Subscription Rcvd:MsgId 0x%x on %s(%d),app %s'
- #define CFE_SB_UNSUB_ARG_ERR_EID 11

  'UnSubscribe Err:Bad Arg,MsgId 0x%x,PipeId %d,app %s,scope %d'
- #define CFE_SB_UNSUB_NO_SUBS_EID 12

  'Unsubscribe Err:No subs for Msg 0x%x on %s,app %s'
- #define CFE_SB_SEND_BAD_ARG_EID 13

  'Send Err:Bad input argument,Arg 0x%x,App %s'
- #define CFE_SB_SEND_NO_SUBS_EID 14

  'No subscribers for MsgId 0x%x,sender %s'
- #define CFE_SB_MSG_TOO_BIG_EID 15

  'Send Err:Msg Too Big MsgId=0x%x,app=%s,size=%d,MaxSz=%d'
- #define CFE_SB_GET_BUF_ERR_EID 16

  'Send Err:Request for Buffer Failed. MsgId 0x%x,app %s,size %d'
- #define CFE_SB_MSGID_LIM_ERR_EID 17

  'Send Err:Msg Limit Err MsgId 0x%x,pipe %s,sender %s'
- #define CFE_SB_RCV_BAD_ARG_EID 18

  'Rcv Err:Bad Input Arg:BufPtr 0x%x,pipe %d,t/o %d,app %s'
- #define CFE_SB_BAD_PIPEID_EID 19

  'Rcv Err:PipeId %d does not exist,app %s'
- #define CFE_SB_DEST_BLK_ERR_EID 20

  'Subscribe Err:Request for Destination Blk failed for Msg 0x%x,Pipe %s'
- #define CFE_SB_SEND_INV_MSGID_EID 21

  'Send Err:Invalid msgid in msg,MsgId 0x%x,App %s'
- #define CFE_SB_SUBSCRIPTION_RPT_EID 22

  'Sending Subscription Report Msg=0x%x,Pipe=%d,Stat=0x%x'
- #define CFE_SB_Q_FULL_ERR_EID 25

  'Pipe Overflow,MsgId 0x%x,pipe %s,stat 0x%x,app %s'
- #define CFE_SB_Q_WR_ERR_EID 26

  'Pipe Write Err,MsgId 0x%x,pipe %s,stat 0x%x,app %s'
- #define CFE_SB_Q_RD_ERR_EID 27

  'Pipe Read Err,pipe %s,app %s,stat 0x%x'
- #define CFE_SB_CMD0_RCVD_EID 28

  'No-op Cmd Rcvd'
- #define CFE_SB_CMD1_RCVD_EID 29

  'Reset Counters Cmd Rcvd'
- #define CFE_SB_LSTSNDER_ERR1_EID 30

  'SB GetLastSender Err:Rcvd Null Ptr,Pipe=d,App=s'
- #define CFE_SB_LSTSNDER_ERR2_EID 31

  'SB GetLastSender Err:Rcvd Invalid Pipe=d,App=s'
- #define CFE_SB_SND_STATS_EID 32

  'Software Bus Statistics packet sent'
- #define CFE_SB_ENBL_RTE1_EID 33

  'Enbl Route Cmd:Route does not exist.Msg 0x%x,Pipe %d'
- #define CFE_SB_ENBL_RTE2_EID 34

  'Enabling Route,Msg 0x%x,Pipe %d'
- #define CFE_SB_ENBL_RTE3_EID 35

'Enbl Route Cmd:Invalid Param.Msg 0x%x,Pipe %d'

- #define CFE_SB_DSBL_RTE1_EID 36

  'Disable Route Cmd:Route does not exist,Msg 0x%x,Pipe %d'
- #define CFE_SB_DSBL_RTE2_EID 37

  'Route Disabled,Msg 0x%x,Pipe %d'
- #define CFE_SB_DSBL_RTE3_EID 38

  'Disable Route Cmd:Invalid Param.Msg 0x%x,Pipe %d'
- #define CFE_SB_SND_RTG_EID 39

  '%s written:Size=%d,Entries=%d'
- #define CFE_SB_SND_RTG_ERR1_EID 40

  'Error creating file %s, stat=0x%x'
- #define CFE_SB_GLS_INV_CALLER_EID 41

  'SB GetLastSender Err:Caller(%s) is not the owner of pipe %d'
- #define CFE_SB_BAD_CMD_CODE_EID 42

  'Invalid Cmd, Unexpected Command Code %d'
- #define CFE_SB_BAD_MSGID_EID 43

  'Invalid Cmd, Unexpected Msg Id: 0x%04x'
- #define CFE_SB_FULL_SUB_PKT_EID 44

  'Full Sub Pkt %d Sent,Entries=%d,Stat=0x%x
  '
- #define CFE_SB_PART_SUB_PKT_EID 45

  'Partial Sub Pkt %d Sent,Entries=%d,Stat=0x%x'
- #define CFE_SB_DEL_PIPE_ERR1_EID 46

  'Pipe Delete Error:Bad Argument,PipedId %d,Requestor %s,Idx %d,Stat %d'
- #define CFE_SB_PIPE_DELETED_EID 47

  'Pipe Deleted:id %d,owner %s'
- #define CFE_SB_SUBSCRIPTION_REMOVED_EID 48

  'Subscription Removed:Msg 0x%x on pipe %d,app %s'
- #define CFE_SB_FILEWRITE_ERR_EID 49

  'File write,byte cnt err,file %s,request=%d,actual=%d'
- #define CFE_SB_SUB_INV_PIPE_EID 50

  'Subscribe Err:Invalid Pipe Id,Msg=0x%x,PipeId=%d,App %s'
- #define CFE_SB_SUB_INV_CALLER_EID 51

  'Subscribe Err:Caller(%s) is not the owner of pipe %d, Msg=0x%x'
- #define CFE_SB_UNSUB_INV_PIPE_EID 52

  'Unsubscribe Err:Invalid Pipe Id Msg=0x%x,Pipe=%d,app=%s'
- #define CFE_SB_UNSUB_INV_CALLER_EID 53

  'Unsubscribe Err:Caller(%s) is not the owner of pipe %d,Msg=0x%x'
- #define CFE_SB_DEL_PIPE_ERR2_EID 54

  'Pipe Delete Error:Caller(%s) is not the owner of pipe %d'
- #define CFE_SB_LEN_ERR_EID 61

  'Invalid cmd length: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'
- #define CFE_SB_CR_PIPE_NAME_TAKEN_EID 62

  'CreatePipeErr:Name Taken:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'
- #define CFE_SB_CR_PIPE_NO_FREE_EID 63

  'CreatePipeErr:No Free:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'

13.61.1 Macro Definition Documentation

13.61.1.1 CFE_SB_BAD_CMD_CODE_EID

```
#define CFE_SB_BAD_CMD_CODE_EID 42
```

Event Message `'Invalid Cmd, Unexpected Command Code %d'`

Type: ERROR

Cause:

This error event message is issued when the SB receives a cmd that has an unexpected cmd code.

Definition at line 738 of file cfe_sb_events.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.61.1.2 CFE_SB_BAD_MSGID_EID

```
#define CFE_SB_BAD_MSGID_EID 43
```

Event Message `'Invalid Cmd, Unexpected Msg Id:  0x%04x'`

Type: ERROR

Cause:

This error event message is issued when the SB receives a msg that has an unexpected msg id.

Definition at line 750 of file cfe_sb_events.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.61.1.3 CFE_SB_BAD_PIPEID_EID

```
#define CFE_SB_BAD_PIPEID_EID 19
```

Event Message `'Rcv Err:PipeId %d does not exist,app %s'`

Type: ERROR

Cause:

This error event message is issued when an invalid PipeId is passed into the CFE_SB_RcvMsg API. The SB Pipe Table shows all valid PipeIds and may be viewed for verification.

Definition at line 458 of file cfe_sb_events.h.

Referenced by CFE_SB_RcvMsg().

13.61.1.4 CFE_SB_CMD0_RCVD_EID

```
#define CFE_SB_CMD0_RCVD_EID 28
```

Event Message `'No-op Cmd Rcvd'`

Type: INFORMATION

Cause:

This info event message is issued in response an SB NO-OP command

Definition at line 557 of file cfe_sb_events.h.

Referenced by CFE_SB_NoopCmd().

13.61.1.5 CFE_SB_CMD1_RCVD_EID

```
#define CFE_SB_CMD1_RCVD_EID 29
```

Event Message `'Reset Counters Cmd Rcvd'`

Type: DEBUG

Cause:

This debug event message is issued in response an SB Reset Counters command

Definition at line 568 of file cfe_sb_events.h.

Referenced by CFE_SB_ResetCountersCmd().

13.61.1.6 CFE_SB_CR_PIPE_BAD_ARG_EID

```
#define CFE_SB_CR_PIPE_BAD_ARG_EID 2
```

Event Message `'CreatePipeErr:Bad Input Arg:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_CreatePipe API receives a bad argument. In this case, a bad argument is defined by the following: A NULL PipeIdPtr, PipeDepth = 0 and PipeDepth > cfg param CFE_PLATFOR↩M_SB_MAX_PIPE_DEPTH

Definition at line 75 of file cfe_sb_events.h.

Referenced by CFE_SB_CreatePipe().

13.61.1.7 CFE_SB_CR_PIPE_ERR_EID

```
#define CFE_SB_CR_PIPE_ERR_EID 4
```

Event Message `'CreatePipeErr:OS_QueueCreate returned %d,app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_CreatePipe API is called and the OS returns an error when the OS returns an error from the OS_QueueCreate API. The error status returned by the OS is displayed in the event. Most commonly, this event is displayed as a result of trying to create pipes with the same name.

Definition at line 102 of file cfe_sb_events.h.

Referenced by CFE_SB_CreatePipe().

13.61.1.8 CFE_SB_CR_PIPE_NAME_TAKEN_EID

```
#define CFE_SB_CR_PIPE_NAME_TAKEN_EID 62
```

Event Message `'CreatePipeErr:Name Taken:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_CreatePipe API tries to create a pipe with a name that is in use.

Definition at line 924 of file cfe_sb_events.h.

Referenced by CFE_SB_CreatePipe().

13.61.1.9 CFE_SB_CR_PIPE_NO_FREE_EID

```
#define CFE_SB_CR_PIPE_NO_FREE_EID 63
```

Event Message `'CreatePipeErr:No Free:app=%s,ptr=0x%x,depth=%d,maxdepth=%d'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_CreatePipe API is unable to create a queue because there are no queues free.

Definition at line 936 of file cfe_sb_events.h.

Referenced by CFE_SB_CreatePipe().

13.61.1.10 CFE_SB_DEL_PIPE_ERR1_EID

```
#define CFE_SB_DEL_PIPE_ERR1_EID 46
```

Event Message `'Pipe Delete Error:Bad Argument,PipedId %d,Requestor %s,Idx %d,Stat %d'`

Type: ERROR

Cause:

This error event message is issued from one of SB's subscribe API's when the function CFE_SB_GetRoutingTblIdx returns an index that is out of range. This error is not expected and is an indication that the SB internal memory has been corrupted.

Definition at line 790 of file cfe_sb_events.h.

Referenced by CFE_SB_DeletePipeFull().

13.61.1.11 CFE_SB_DEL_PIPE_ERR2_EID

```
#define CFE_SB_DEL_PIPE_ERR2_EID 54
```

Event Message `'Pipe Delete Error:Caller(%s) is not the owner of pipe %d'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_DeletePipe API is called by a task that is not the owner of the pipe. Pipes may be deleted only by the task that created the pipe or ES(for cleanup purposes).

Definition at line 894 of file cfe_sb_events.h.

Referenced by CFE_SB_DeletePipeFull().

13.61.1.12 CFE_SB_DEST_BLK_ERR_EID

```
#define CFE_SB_DEST_BLK_ERR_EID 20
```

Event Message `'Subscribe Err:Request for Destination Blk failed for Msg 0x%x,Pipe %s'`

Type: ERROR

Cause:

This error event message is issued when the SB receives an error from the memory pool in the attempt to obtain a new destination block. Then memory pool statistics may be viewed by sending the related ES command.

Definition at line 472 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.13 CFE_SB_DSBL_RTE1_EID

```
#define CFE_SB_DSBL_RTE1_EID 36
```

Event Message `'Disable Route Cmd:Route does not exist,Msg 0x%x,Pipe %d'`

Type: ERROR

Cause:

This error event message is issued when SB receives a cmd to disable a route that does not exist in the routing table. A route is defined by a MsgId, PipeId pair.

Definition at line 657 of file cfe_sb_events.h.

Referenced by CFE_SB_DisableRouteCmd().

13.61.1.14 CFE_SB_DSBL_RTE2_EID

```
#define CFE_SB_DSBL_RTE2_EID 37
```

Event Message `'Route Disabled,Msg 0x%x,Pipe %d'`

Type: DEBUG

Cause:

This debug event message is issued when SB receives a cmd to disable a route and the request is successfully executed.

Definition at line 669 of file cfe_sb_events.h.

Referenced by CFE_SB_DisableRouteCmd().

13.61.1.15 CFE_SB_DSBL_RTE3_EID

```
#define CFE_SB_DSBL_RTE3_EID 38
```

Event Message `'Disable Route Cmd:Invalid Param.Msg 0x%x,Pipe %d'`

Type: ERROR

Cause:

This error event message is issued when SB receives a cmd to disable a route and the MsgId or PipeId does not pass the validation checks. The MsgId must be less than cfg param CFE_PLATFORM_SB_HIGHEST_VALID_MSGID. The PipeId must exist and be less than cfg param CFE_PLATFORM_SB_MAX_PIPES. The SB pipe table may be viewed to verify the PipeId existence.

Definition at line 684 of file cfe_sb_events.h.

Referenced by CFE_SB_DisableRouteCmd().

13.61.1.16 CFE_SB_DUP_SUBSCRIP_EID

```
#define CFE_SB_DUP_SUBSCRIP_EID 7
```

Event Message `'Duplicate Subscription,MsgId 0x%x on %s pipe,app %s'`

Type: INFORMATION

Cause:

This info event message is issued when a subscription request is received that already exists in the routing table. A duplicate subscription is defined by a matching MsgId and PipeId. No other parameters are used in detecting a duplicate subscription. NOTE: By default, SB filters this event. The EVS filter algorithm allows the first event to pass through the filter, but all subsequent events with this event id will be filtered. A command must be sent to unfilter this event if the user desires to see it.

Definition at line 284 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.17 CFE_SB_ENBL_RTE1_EID

```
#define CFE_SB_ENBL_RTE1_EID 33
```

Event Message `'Enbl Route Cmd:Route does not exist.Msg 0x%x,Pipe %d'`

Type: ERROR

Cause:

This error event message is issued when SB receives a cmd to enable a route that does not exist in the routing table. A route is defined by a MsgId, PipeId pair.

Definition at line 618 of file cfe_sb_events.h.

Referenced by CFE_SB_EnableRouteCmd().

13.61.1.18 CFE_SB_ENBL_RTE2_EID

```
#define CFE_SB_ENBL_RTE2_EID 34
```

Event Message `'Enabling Route,Msg 0x%x,Pipe %d'`

Type: DEBUG

Cause:

This debug event message is issued when SB receives a cmd to enable a route and the request is successfully executed.

Definition at line 630 of file cfe_sb_events.h.

Referenced by CFE_SB_EnableRouteCmd().

13.61.1.19 CFE_SB_ENBL_RTE3_EID

```
#define CFE_SB_ENBL_RTE3_EID 35
```

Event Message `'Enbl Route Cmd:Invalid Param.Msg 0x%x,Pipe %d'`

Type: ERROR

Cause:

This error event message is issued when SB receives a cmd to enable a route and the MsgId or PipeId does not pass the validation checks. The MsgId must be less than cfg param CFE_PLATFORM_SB_HIGHEST_VALID_MSGID. The PipeId must exist and be less than cfg param CFE_PLATFORM_SB_MAX_PIPES. The SB pipe table may be viewed to verify the PipeId existence.

Definition at line 645 of file cfe_sb_events.h.

Referenced by CFE_SB_EnableRouteCmd().

13.61.1.20 CFE_SB_FILEWRITE_ERR_EID

```
#define CFE_SB_FILEWRITE_ERR_EID 49
```

Event Message `'File write,byte cnt err,file %s,request=%d,actual=%d'`

Type: ERROR

Cause:

This error event message is issued when one of many SB's file write operations is unsuccessful. This event is a result of CFE_FS_WriteHeader or OS_write returning something other than the number of bytes requested to be written. The requested value and the return value are displayed in the event.

Definition at line 828 of file cfe_sb_events.h.

Referenced by CFE_SB_FileWriteByteCntErr().

13.61.1.21 CFE_SB_FULL_SUB_PKT_EID

```
#define CFE_SB_FULL_SUB_PKT_EID 44
```

Event Message `'Full Sub Pkt %d Sent,Entries=%d,Stat=0x%x
'`

Type: DEBUG

Cause:

This debug event message is issued in response to the 'Send Previous Subscriptions' command and a full pkt segment is sent.

Definition at line 763 of file cfe_sb_events.h.

Referenced by CFE_SB_SendPrevSubsCmd().

13.61.1.22 CFE_SB_GET_BUF_ERR_EID

```
#define CFE_SB_GET_BUF_ERR_EID 16
```

Event Message `'Send Err:Request for Buffer Failed.  MsgId 0x%x,app %s,size %d'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API fails to receive the necessary buffer memory from the ES memory pool. This could be an indication that the cfg param CFE_PLATFORM_SB_BUF_MEMORY_BYTES is set too low. To check this, send SB cmd to dump the SB statistics pkt and view the buffer memory parameters.

Definition at line 413 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.23 CFE_SB_GETPIPEIDBYNAME_EID

```
#define CFE_SB_GETPIPEIDBYNAME_EID 65
```

Event Message `'GetPipeIdByName:  ID retrieved.  Name %s,IdOut 0x%x, app %s'`

Type: DEBUG

Cause:

This debug event is generated when id is retrieved by name.

Definition at line 228 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeIdByName().

13.61.1.24 CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID

```
#define CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID 67
```

Event Message `'GetPipeIdByName Err:Name not found,Name %s,IdOut 0xx,App %s'`

Type: ERROR

Cause:

This error event message is issued when the #CFE_SB_GetMsgIdByName API receives an invalid name.

Definition at line 252 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeIdByName().

13.61.1.25 CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID

```
#define CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID 66
```

Event Message `'GetPipeIdByName Err:Bad input argument,Name 0x%x,IdOut 0xx,App %s'`

Type: ERROR

Cause:

This error event message is issued when the #CFE_SB_GetMsgIdByName API receives an invalid (possibly NULL) ptr as an argument.

Definition at line 240 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeIdByName().

13.61.1.26 CFE_SB_GETPIPENAME_EID

```
#define CFE_SB_GETPIPENAME_EID 62
```

Event Message `'GetPipeName:  Name retrieved.  NameOut %s,Id %d, app %s'`

Type: DEBUG

Cause:

This debug event is generated when name is retrieved by id.

Definition at line 195 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeName().

13.61.1.27 CFE_SB_GETPIPENAME_ID_ERR_EID

```
#define CFE_SB_GETPIPENAME_ID_ERR_EID 64
```

Event Message `'GetPipeName: Id error. NameOut %s,Id %d, app %s'`

Type: ERROR

Cause:

This debug event is generated when name is retrieved by id.

Definition at line 217 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeName().

13.61.1.28 CFE_SB_GETPIPENAME_NULL_PTR_EID

```
#define CFE_SB_GETPIPENAME_NULL_PTR_EID 63
```

Event Message `'GetPipeName: Null ptr error. Id %d, app %s'`

Type: ERROR

Cause:

This debug event is generated when the name buffer ptr is null.

Definition at line 206 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeName().

13.61.1.29 CFE_SB_GETPIPEOPTS_EID

```
#define CFE_SB_GETPIPEOPTS_EID 60
```

Event Message `'GetPipeOpts:  Options retrieved.  app %s'`

Type: DEBUG

Cause:

This debug event is generated when options are retrieved.

Definition at line 184 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeOpts().

13.61.1.30 CFE_SB_GETPIPEOPTS_ID_ERR_EID

```
#define CFE_SB_GETPIPEOPTS_ID_ERR_EID 58
```

Event Message `'GetPipeOptsErr:Invalid pipe id (%d).app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_GetPipeOpts API is called and the PipeID is invalid.

Definition at line 161 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeOpts().

13.61.1.31 CFE_SB_GETPIPEOPTS_PTR_ERR_EID

```
#define CFE_SB_GETPIPEOPTS_PTR_ERR_EID 59
```

Event Message `'GetPipeOptsErr:Invalid opts ptr.app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_GetPipeOpts API is called and the pointer is invalid.

Definition at line 173 of file cfe_sb_events.h.

Referenced by CFE_SB_GetPipeOpts().

13.61.1.32 CFE_SB_GLS_INV_CALLER_EID

```
#define CFE_SB_GLS_INV_CALLER_EID 41
```

Event Message `'SB GetLastSender Err:Caller(%s) is not the owner of pipe %d'`

Type: ERROR

Cause:

This error event message is issued when the caller of CFE_SB_GetLastSenderId is not the owner of the given pipe Id.

Definition at line 725 of file cfe_sb_events.h.

Referenced by CFE_SB_GetLastSenderId().

13.61.1.33 CFE_SB_INIT_EID

```
#define CFE_SB_INIT_EID 1
```

Event Message `'cFE SB Initialized'`

Type: INFORMATION

Cause:

This event message is issued when the Software Bus Task completes its initialization.

Definition at line 62 of file cfe_sb_events.h.

Referenced by CFE_SB_AppInit().

13.61.1.34 CFE_SB_LEN_ERR_EID

```
#define CFE_SB_LEN_ERR_EID 61
```

Event Message `'Invalid cmd length:  ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_SB_CMD_MID message ID has arrived but whose packet length does not match the expected length for the specified command code.

The `ID` field in the event message specifies the Message ID (in hex), the `CC` field specifies the Command Code (in decimal), the `Exp` Len field specified the Expected Length (in decimal), and `Len` specifies the message Length (in decimal) found in the message.

Definition at line 912 of file cfe_sb_events.h.

Referenced by CFE_SB_VerifyCmdLength().

13.61.1.35 CFE_SB_LSTSNDER_ERR1_EID

```
#define CFE_SB_LSTSNDER_ERR1_EID 30
```

Event Message `'SB GetLastSender Err:Rcvd Null Ptr,Pipe=d,App=s'`

Type: ERROR

Cause:

This error event message is issued when SB receives a Null pointer from the caller of CFE_SB_GetLastSenderId.

Definition at line 581 of file cfe_sb_events.h.

Referenced by CFE_SB_GetLastSenderId().

13.61.1.36 CFE_SB_LSTSNDER_ERR2_EID

```
#define CFE_SB_LSTSNDER_ERR2_EID 31
```

Event Message `'SB GetLastSender Err:Rcvd Invalid Pipe=d,App=s'`

Type: ERROR

Cause:

This error event message is issued when SB receives an invalid pipe from the caller of CFE_SB_GetLastSenderId.

Definition at line 593 of file cfe_sb_events.h.

Referenced by CFE_SB_GetLastSenderId().

13.61.1.37 CFE_SB_MAX_DESTS_MET_EID

```
#define CFE_SB_MAX_DESTS_MET_EID 9
```

Event Message `'Subscribe Err:Max Dests(%d)In Use For Msg 0x%x,pipe %s,app %s'`

Type: ERROR

Cause:

This error event message is issued when a subscription request is received and all destinations for that MsgId are in use. The number of destinations per msgid is a configuration parameter named CFE_PLATFORM_SB_MAX_DEST↩ _PER_PKT. A destination is defined as a pipe.

Definition at line 315 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.38 CFE_SB_MAX_EID

```
#define CFE_SB_MAX_EID 67
```

Definition at line 43 of file cfe_sb_events.h.

13.61.1.39 CFE_SB_MAX_MSGS_MET_EID

```
#define CFE_SB_MAX_MSGS_MET_EID 8
```

Event Message `'Subscribe Err:Max Msgs(%d)In Use,MsgId 0x%x,pipe %s,app %s'`

Type: ERROR

Cause:

This error event message is issued when one of the SB subscribe APIs are called with a new MsgId, and SB cannot accommodate the new MsgId because the maximum number of MsgIds are in use. The maximum number of MsgIds is defined by cfg param CFE_PLATFORM_SB_MAX_MSG_IDS. This cfg param dictates the number of elements in the SB routing table. There is one element per MsgId. The user may monitor the routing table utilization figures (msgids currently in use, high water mark and max allowed) by sending the SB cmd to dump the SB statistics data.

Definition at line 301 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.40 CFE_SB_MAX_PIPES_MET_EID

```
#define CFE_SB_MAX_PIPES_MET_EID 3
```

Event Message `'CreatePipeErr:Max Pipes(%d)In Use.app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_CreatePipe API is called and the maximum number of pipes (defined by cfg param CFE_PLATFORM_SB_MAX_PIPES) are in use.

Definition at line 87 of file cfe_sb_events.h.

Referenced by CFE_SB_CreatePipe().

13.61.1.41 CFE_SB_MSG_TOO_BIG_EID

```
#define CFE_SB_MSG_TOO_BIG_EID 15
```

Event Message `'Send Err:Msg Too Big MsgId=0x%x,app=%s,size=%d,MaxSz=%d'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API is called and the packet length field in the message header implies that the message size exceeds the max size defined by mission cfg param CFE_MISSION_SB_MAX↩_SB_MSG_SIZE. The request to send the message is denied, there is no partial packet sent.

Definition at line 399 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.42 CFE_SB_MSGID_LIM_ERR_EID

```
#define CFE_SB_MSGID_LIM_ERR_EID 17
```

Event Message `'Send Err:Msg Limit Err MsgId 0x%x,pipe %s,sender %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API cannot route the MsgId (displayed in event) to the pipe (displayed in the event) because the pipe currently contains the maximum number of messages of this type (MsgId). This is typically an indication that the receiver is not reading its pipe fast enough, or at all. A less typical scenerio is that the sender is sending a burst of pkts of this type (or MsgId) and the receiver (owner of 'pipe') cannot keep up. The subscriber of the message dictates this limit count in the 'MsgLim' parameter of the CFE_SB_SubscribeEx API or uses the default value of 4 if using the CFE_SB_Subscribe API.

Definition at line 432 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.43 CFE_SB_PART_SUB_PKT_EID

```
#define CFE_SB_PART_SUB_PKT_EID 45
```

Event Message `'Partial Sub Pkt %d Sent,Entries=%d,Stat=0x%x'`

Type: DEBUG

Cause:

This debug event message is issued in response to the 'Send Previous Subscriptions' command and a partial pkt segment is sent.

Definition at line 775 of file cfe_sb_events.h.

Referenced by CFE_SB_SendPrevSubsCmd().

13.61.1.44 CFE_SB_PIPE_ADDED_EID

```
#define CFE_SB_PIPE_ADDED_EID 5
```

Event Message `'Pipe Created:name %s,id %d,app %s'`

Type: DEBUG

Cause:

This debug event message is issued when a pipe was successfully created in the CFE_SB_CreatePipe API.

Definition at line 114 of file cfe_sb_events.h.

Referenced by CFE_SB_CreatePipe().

13.61.1.45 CFE_SB_PIPE_DELETED_EID

```
#define CFE_SB_PIPE_DELETED_EID 47
```

Event Message `'Pipe Deleted:id %d,owner %s'`

Type: DEBUG

Cause:

This debug event message is issued when the CFE_SB_DeletePipe API is called and the request is successfully completed.

Definition at line 802 of file cfe_sb_events.h.

Referenced by CFE_SB_DeletePipeFull().

13.61.1.46 CFE_SB_Q_FULL_ERR_EID

```
#define CFE_SB_Q_FULL_ERR_EID 25
```

Event Message `'Pipe Overflow,MsgId 0x%x,pipe %s,stat 0x%x,app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API is called and encounters an error when attempting to write the msg to the destination pipe (which is an underlying queue). This could indicate that the owner of the pipe is not readings its messages fast enough or at all. It may also mean that the pipe depth is not deep enough. The pipe depth is an input parameter to the CFE_SB_CreatePipe API.

Definition at line 514 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.47 CFE_SB_Q_RD_ERR_EID

```
#define CFE_SB_Q_RD_ERR_EID 27
```

Event Message `'Pipe Read Err,pipe %s,app %s,stat 0x%x'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API is called and encounters an error when attempting to read the msg from the destination pipe (which is an underlying queue). More precisely, the OS API OS_QueueGet has returned an unexpected error. The return code is displayed in the event. For more information, the user may look up the return code in the OSAL documention or source code.

Definition at line 546 of file cfe_sb_events.h.

Referenced by CFE_SB_ReadQueue().

13.61.1.48 CFE_SB_Q_WR_ERR_EID

```
#define CFE_SB_Q_WR_ERR_EID 26
```

Event Message `'Pipe Write Err,MsgId 0x%x,pipe %s,stat 0x%x,app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API is called and encounters an error when attempting to write the msg to the destination pipe (which is an underlying queue). More precisely, the OS API OS_QueuePut has returned an unexpected error. The return code is displayed in the event. For more information, the user may look up the return code in the OSAL documention or source code.

Definition at line 530 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.49 CFE_SB_RCV_BAD_ARG_EID

```
#define CFE_SB_RCV_BAD_ARG_EID 18
```

Event Message `'Rcv Err:Bad Input Arg:BufPtr 0x%x,pipe %d,t/o %d,app %s'`

Type: ERROR

Cause:

This error event message is issued when an invalid paramter is passed into the CFE_SB_RcvMsg API. Two possibile problems would be the first parameter (∗BufPtr) being NULL or the third paramter (TimeOut) being less than -1.

Definition at line 445 of file cfe_sb_events.h.

Referenced by CFE_SB_RcvMsg().

13.61.1.50 CFE_SB_SEND_BAD_ARG_EID

```
#define CFE_SB_SEND_BAD_ARG_EID 13
```

Event Message `'Send Err:Bad input argument,Arg 0x%x,App %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API receives an invalid (possibly NULL) ptr as an argument.

Definition at line 367 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.51 CFE_SB_SEND_INV_MSGID_EID

```
#define CFE_SB_SEND_INV_MSGID_EID 21
```

Event Message `'Send Err:Invalid msgid in msg,MsgId 0x%x,App %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SendMsg API is called and the SB discovers that the message to send has a msg id that is invalid. It may be due to a msg id that is greater than cfg parameter CFE_PLATFORM_S↩B_HIGHEST_VALID_MSGID

Definition at line 486 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.52 CFE_SB_SEND_NO_SUBS_EID

```
#define CFE_SB_SEND_NO_SUBS_EID 14
```

Event Message `'No subscribers for MsgId 0x%x,sender %s'`

Type: INFORMATION

Cause:

This info event message is issued when the CFE_SB_SendMsg API is called and there are no subscribers (therefore no destinations) for the message to be sent. Each time the SB detects this situation, the corresponding SB telemetry point is incremented.. NOTE: By default, SB filters this event. The EVS filter algorithm allows the first event to pass through the filter, but all subsequent events with this event id will be filtered. A command must be sent to unfilter this event if the user desires to see it.

Definition at line 385 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMsgFull().

13.61.1.53 CFE_SB_SETPIPEOPTS_EID

```
#define CFE_SB_SETPIPEOPTS_EID 57
```

Event Message `'SetPipeOpts: Options set (%d). app %s'`

Type: DEBUG

Cause:

This debug event is generated when options are set.

Definition at line 149 of file cfe_sb_events.h.

Referenced by CFE_SB_SetPipeOpts().

13.61.1.54 CFE_SB_SETPIPEOPTS_ID_ERR_EID

```
#define CFE_SB_SETPIPEOPTS_ID_ERR_EID 55
```

Event Message `'SetPipeOptsErr:Invalid pipe id (%d).app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SetPipeOpts API is called and the PipeID is invalid.

Definition at line 126 of file cfe_sb_events.h.

Referenced by CFE_SB_SetPipeOpts().

13.61.1.55 CFE_SB_SETPIPEOPTS_OWNER_ERR_EID

```
#define CFE_SB_SETPIPEOPTS_OWNER_ERR_EID 56
```

Event Message `'SetPipeOptsErr:Caller not owner (%d).app %s'`

Type: ERROR

Cause:

This error event message is issued when the CFE_SB_SetPipeOpts API is called and the pipe is owned by another app ID.

Definition at line 138 of file cfe_sb_events.h.

Referenced by CFE_SB_SetPipeOpts().

13.61.1.56 CFE_SB_SND_RTG_EID

```
#define CFE_SB_SND_RTG_EID 39
```

Event Message `'%s written:Size=%d,Entries=%d'`

Type: DEBUG

Cause:

This debug event message is issued after the SB routing info file, pipe info file or the map info file is written and closed. This is done is response to the SB 'Send Routing Info' cmd, the SB 'Send pipe Info' cmd or the SB 'Send Map Info' cmd, respectively.

Definition at line 698 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.61.1.57 CFE_SB_SND_RTG_ERR1_EID

```
#define CFE_SB_SND_RTG_ERR1_EID 40
```

Event Message `'Error creating file %s, stat=0x%x'`

Type: ERROR

Cause:

This error event message is issued when the SB 'Send Routing Info' cmd is received and the file create fails. The event displays the status received from the OS.

Definition at line 712 of file cfe_sb_events.h.

Referenced by CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.61.1.58 CFE_SB_SND_STATS_EID

```
#define CFE_SB_SND_STATS_EID 32
```

Event Message `'Software Bus Statistics packet sent'`

Type: DEBUG

Cause:

This debug event message is issued when SB receives a cmd to send the SB statistics pkt.

Definition at line 606 of file cfe_sb_events.h.

Referenced by CFE_SB_SendStatsCmd().

13.61.1.59 CFE_SB_SUB_ARG_ERR_EID

```
#define CFE_SB_SUB_ARG_ERR_EID 6
```

Event Message `'Subscribe Err:Bad Arg,MsgId 0x%x,PipeId %d,app %s,scope %d'`

Type: ERROR

Cause:

This error event message is issued when one of the Subscribe API's are called with an invalid MsgId. An invalid MsgId is defined as being greater than the cfg param CFE_PLATFORM_SB_HIGHEST_VALID_MSGID.

Definition at line 266 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.60 CFE_SB_SUB_INV_CALLER_EID

```
#define CFE_SB_SUB_INV_CALLER_EID 51
```

Event Message `'Subscribe Err:Caller(%s) is not the owner of pipe %d, Msg=0x%x'`

Type: ERROR

Cause:

This error event message is issued when one of the SB subscribe API's are called and the requestor is not the owner of the pipe. Only the owner of the pipe may subscribe to messages on the pipe.

Definition at line 854 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.61 CFE_SB_SUB_INV_PIPE_EID

```
#define CFE_SB_SUB_INV_PIPE_EID 50
```

Event Message `'Subscribe Err:Invalid Pipe Id,Msg=0x%x,PipeId=%d,App %s'`

Type: ERROR

Cause:

This error event message is issued when the input PipeId has a value that is not listed in the pipe table. This typically means that the pipe does not exist. The pipe table may be viewed for verification.

Definition at line 841 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.62 CFE_SB_SUBSCRIPTION_RCVD_EID

```
#define CFE_SB_SUBSCRIPTION_RCVD_EID 10
```

Event Message `'Subscription Rcvd:MsgId 0x%x on %s(%d),app %s'`

Type: DEBUG

Cause:

This debug event message is issued when a subscription is successfully made through one of the SB Subscribe API's

Definition at line 327 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.63 CFE_SB_SUBSCRIPTION_REMOVED_EID

```
#define CFE_SB_SUBSCRIPTION_REMOVED_EID 48
```

Event Message `'Subscription Removed:Msg 0x%x on pipe %d,app %s'`

Type: DEBUG

Cause:

This debug event message is issued when CFE_SB_Unsubscribe API is called and the request is successfully completed.

Definition at line 814 of file cfe_sb_events.h.

Referenced by CFE_SB_UnsubscribeFull().

13.61.1.64 CFE_SB_SUBSCRIPTION_RPT_EID

```
#define CFE_SB_SUBSCRIPTION_RPT_EID 22
```

Event Message `'Sending Subscription Report Msg=0x%x,Pipe=%d,Stat=0x%x'`

Type: DEBUG

Cause:

This debug event message is issued when SB subscription reporting is enabled, (which is disabled by default) and a subscription is successfully received.

Definition at line 498 of file cfe_sb_events.h.

Referenced by CFE_SB_SubscribeFull().

13.61.1.65 CFE_SB_UNSUB_ARG_ERR_EID

```
#define CFE_SB_UNSUB_ARG_ERR_EID 11
```

Event Message `'UnSubscribe Err:Bad Arg,MsgId 0x%x,PipeId %d,app %s,scope %d'`

Type: ERROR

Cause:

This error event message is issued when a request to unsubscribe fails due to an invalid msgid or an invalid pipeid in one of SB's unsubscribe API's. The msgid must be less than cfg param CFE_PLATFORM_SB_HIGHEST_VALID_MSGID and the pipeid must have been created and have a value less than cfg param CFE_PLATFORM_SB_MAX_PIPES. The SB pipe table may be viewed to verify its value or existence.

Definition at line 342 of file cfe_sb_events.h.

Referenced by CFE_SB_UnsubscribeFull().

13.61.1.66 CFE_SB_UNSUB_INV_CALLER_EID

```
#define CFE_SB_UNSUB_INV_CALLER_EID 53
```

Event Message `'Unsubscribe Err:Caller(%s) is not the owner of pipe %d,Msg=0x%x'`

Type: ERROR

Cause:

This error event message is issued when one of the SB unsubscribe API's are called and the requestor is not the owner of the pipe (or ES). Only the owner of the pipe(or ES for cleanup purposes)may unsubscribe messages from a pipe.

Definition at line 881 of file cfe_sb_events.h.

Referenced by CFE_SB_UnsubscribeFull().

13.61.1.67 CFE_SB_UNSUB_INV_PIPE_EID

```
#define CFE_SB_UNSUB_INV_PIPE_EID 52
```

Event Message `'Unsubscribe Err:Invalid Pipe Id Msg=0x%x,Pipe=%d,app=%s'`

Type: ERROR

Cause:

This error event message is issued when one of the SB unsubscribe API's are called and the input parameter PipeId is not listed in the pipe table. This typically means that the pipe does not exist. The pipe table may be viewed for verification.

Definition at line 868 of file cfe_sb_events.h.

Referenced by CFE_SB_UnsubscribeFull().

13.61.1.68 CFE_SB_UNSUB_NO_SUBS_EID

```
#define CFE_SB_UNSUB_NO_SUBS_EID 12
```

Event Message `'Unsubscribe Err:No subs for Msg 0x%x on %s,app %s'`

Type: INFORMATION

Cause:

This info event message is issued when a request to unsubscribe fails due to a non existent msgid/pipeid combination in the SB routing table. The SB routing table may be viewed to see a list of valid msgid/pipeid combinations.

Definition at line 355 of file cfe_sb_events.h.

Referenced by CFE_SB_UnsubscribeFull().

13.62 cfe/fsw/cfe-core/src/inc/cfe_sb_extern_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
```

Typedefs

- typedef uint8 CFE_SB_QosPriority_Enum_t
  Selects the priorty level for message routing.
- typedef uint8 CFE_SB_QosReliability_Enum_t
  Selects the reliability level for message routing.
- typedef uint16 CFE_SB_MsgRouteIdx_Atom_t
  An integer type that should be used for indexing into the Routing Table.
- typedef uint16 CFE_SB_MsgId_Atom_t
  CFE_SB_MsgId_Atom_t primitive type definition.
- typedef CFE_SB_MsgId_Atom_t CFE_SB_MsgId_t
  CFE_SB_MsgId_t type definition.

Enumerations

- enum CFE_SB_QosPriority { CFE_SB_QosPriority_LOW = 0, CFE_SB_QosPriority_HIGH = 1 }
  Label definitions associated with CFE_SB_QosPriority_Enum_t.
- enum CFE_SB_QosReliability { CFE_SB_QosReliability_LOW = 0, CFE_SB_QosReliability_HIGH = 1 }
  Label definitions associated with CFE_SB_QosReliability_Enum_t.

13.62.1 Typedef Documentation

13.62.1.1 CFE_SB_MsgId_Atom_t

```
typedef uint16 CFE_SB_MsgId_Atom_t
```

This is an integer type capable of holding any Message ID value

Definition at line 101 of file cfe_sb_extern_typedefs.h.

13.62.1.2 CFE_SB_MsgId_t

```
typedef CFE_SB_MsgId_Atom_t CFE_SB_MsgId_t
```

Software Bus message identifier used in many SB APIs

Currently this is directly mapped to the underlying holding type (not wrapped) for compatibility with existing usage semantics in apps (mainly switch/case statements)

Note

In a future version it could become a type-safe wrapper similar to the route index, to avoid message IDs getting mixed between other integer values.

Definition at line 115 of file cfe_sb_extern_typedefs.h.

13.62.1.3 CFE_SB_MsgRouteIdx_Atom_t

```
typedef uint16 CFE_SB_MsgRouteIdx_Atom_t
```

Definition at line 91 of file cfe_sb_extern_typedefs.h.

13.62.1.4 CFE_SB_QosPriority_Enum_t

```
typedef uint8 CFE_SB_QosPriority_Enum_t
```

See also

enum CFE_SB_QosPriority

Definition at line 60 of file cfe_sb_extern_typedefs.h.

13.62.1.5 CFE_SB_QosReliability_Enum_t

```
typedef uint8 CFE_SB_QosReliability_Enum_t
```

See also

enum CFE_SB_QosReliability

Definition at line 86 of file cfe_sb_extern_typedefs.h.

13.62.2 Enumeration Type Documentation

13.62.2.1 CFE_SB_QosPriority

```
enum CFE_SB_QosPriority
```

Enumerator

CFE_SB_QosPriority_LOW	Normal priority level.
CFE_SB_QosPriority_HIGH	High priority.

Definition at line 40 of file cfe_sb_extern_typedefs.h.

13.62.2.2 CFE_SB_QosReliability

```
enum CFE_SB_QosReliability
```

Enumerator

CFE_SB_QosReliability_LOW	Normal (best-effort) reliability.
CFE_SB_QosReliability_HIGH	High reliability.

Definition at line 66 of file cfe_sb_extern_typedefs.h.

13.63 cfe/fsw/cfe-core/src/inc/cfe_sb_msg.h File Reference

```
#include "common_types.h"
#include "cfe_sb.h"
#include "cfe_es.h"
```

Data Structures

- struct CFE_SB_WriteFileInfoCmd_Payload_t

  Write File Info Commands.
- struct CFE_SB_WriteFileInfoCmd_t
- struct CFE_SB_RouteCmd_Payload_t

  Enable/Disable Route Commands.
- struct CFE_SB_RouteCmd_t
- struct CFE_SB_HousekeepingTlm_Payload_t
- struct CFE_SB_HousekeepingTlm_t
- struct CFE_SB_PipeDepthStats_t

  SB Pipe Depth Statistics.
- struct CFE_SB_StatsTlm_Payload_t
- struct CFE_SB_StatsTlm_t
- struct CFE_SB_RoutingFileEntry_t

  SB Routing File Entry.
- struct CFE_SB_MsgMapFileEntry_t

  SB Map File Entry.
- struct CFE_SB_SingleSubscriptionTlm_Payload_t
- struct CFE_SB_SingleSubscriptionTlm_t
- struct CFE_SB_SubEntries_t

  SB Previous Subscriptions Entry.
- struct CFE_SB_AllSubscriptionsTlm_Payload_t
- struct CFE_SB_AllSubscriptionsTlm_t

Macros

- #define CFE_SB_NOOP_CC 0
- #define CFE_SB_RESET_COUNTERS_CC 1
- #define CFE_SB_SEND_SB_STATS_CC 2
- #define CFE_SB_SEND_ROUTING_INFO_CC 3
- #define CFE_SB_ENABLE_ROUTE_CC 4
- #define CFE_SB_DISABLE_ROUTE_CC 5
- #define CFE_SB_SEND_PIPE_INFO_CC 7
- #define CFE_SB_SEND_MAP_INFO_CC 8
- #define CFE_SB_ENABLE_SUB_REPORTING_CC 9
- #define CFE_SB_DISABLE_SUB_REPORTING_CC 10
- #define CFE_SB_SEND_PREV_SUBS_CC 11

Typedefs

- typedef CFE_SB_CmdHdr_t CFE_SB_Noop_t
- typedef CFE_SB_CmdHdr_t CFE_SB_ResetCounters_t
- typedef CFE_SB_CmdHdr_t CFE_SB_EnableSubReporting_t
- typedef CFE_SB_CmdHdr_t CFE_SB_DisableSubReporting_t
- typedef CFE_SB_CmdHdr_t CFE_SB_SendSbStats_t
- typedef CFE_SB_CmdHdr_t CFE_SB_SendPrevSubs_t
- typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_SendRoutingInfo_t

- typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_SendPipeInfo_t
- typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_SendMapInfo_t
- typedef CFE_SB_RouteCmd_t CFE_SB_EnableRoute_t
- typedef CFE_SB_RouteCmd_t CFE_SB_DisableRoute_t
- typedef CFE_SB_HousekeepingTlm_t CFE_SB_HKMsg_t
- typedef CFE_SB_StatsTlm_t CFE_SB_StatMsg_t
- typedef CFE_SB_AllSubscriptionsTlm_t CFE_SB_PrevSubMsg_t
- typedef CFE_SB_SingleSubscriptionTlm_t CFE_SB_SubRprtMsg_t

13.63.1 Macro Definition Documentation

13.63.1.1 CFE_SB_DISABLE_ROUTE_CC

```
#define CFE_SB_DISABLE_ROUTE_CC 5
```

Name Disable Software Bus Route

Description

This command will disable a particular destination. The destination is specified in terms of MsgID and PipeID. The MsgId and PipeID are parmaters in the command. All destinations are enabled by default.

Command Mnemonic(s) **`$sc_$cpu_SB_DisRoute`**

Command Structure

CFE_SB_RouteCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment
- View routing information CFE_SB_SEND_ROUTING_INFO_CC to verify enable/disable state change
- The CFE_SB_DSBL_RTE2_EID debug event message will be generated. All debug events are filtered by default.
- Destination will stop receiving messages.

Error Conditions

An Error may occur if the MsgId or PipeId parmaters do not pass validation or the destination does not exist.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_SB_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases. See CFE_SB_DSBL_RTE1_EID or CFE_SB_DSBL_RTE3_EID

Criticality

This command is not intended to be used in nominal conditions. It is possible to get into a state where a destination cannot be re-enabled without reseting the processor. For instance, sending this command with CFE_SB_CMD_MID and the SB_Cmd_Pipe would inhibit any ground commanding to the software bus until the processor was reset. There are similar problems that may occur when using this command.

See also

CFE_SB_SEND_ROUTING_INFO_CC, CFE_SB_ENABLE_ROUTE_CC, CFE_SB_RouteCmd_t

Definition at line 277 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.2 CFE_SB_DISABLE_SUB_REPORTING_CC

```
#define CFE_SB_DISABLE_SUB_REPORTING_CC 10
```

Name Disable Subscription Reporting Command

This command will disable subscription reporting and is intended to

be used only by the CFS SBN (Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When subscription reporting is enabled, SB will generate and send a software bus packet for each subscription received. The software bus packet that is sent contains the information received in the subscription API. This subscription report is neeeded by SBN if offboard routing is required.

Command Mnemonic(s) **`$sc_$cpu_SB_DisSubRptg`**

Command Structure

CFE_SB_CmdHdr_t

Command Verification

Successful execution of this command will result in the suppression of packets (with the CFE_SB_ONESUB_TL↩M_MID MsgId) for each subscription received by SB through the subscription APIs.

Error Conditions

None

Criticality

None

See also

CFE_SB_SingleSubscriptionTlm_t, CFE_SB_ENABLE_SUB_REPORTING_CC, CFE_SB_SEND_PREV_SUB↩S_CC

Definition at line 430 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.3 CFE_SB_ENABLE_ROUTE_CC

```
#define CFE_SB_ENABLE_ROUTE_CC 4
```

Name Enable Software Bus Route

Description

This command will enable a particular destination. The destination is specified in terms of MsgID and PipeID. The MsgId and PipeID are parmaters in the command. All destinations are enabled by default. This command is needed only after a CFE_SB_DISABLE_ROUTE_CC command is used.

Command Mnemonic(s) **`$sc_$cpu_SB_EnaRoute`**

Command Structure

CFE_SB_RouteCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment
- View routing information CFE_SB_SEND_ROUTING_INFO_CC to verify enable/disable state change
- The CFE_SB_ENBL_RTE2_EID debug event message will be generated. All debug events are filtered by default.
- Destination will begin receiving messages.

Error Conditions

An Error may occur if the MsgId or PipeId parmaters do not pass validation or the destination does not exist.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_SB_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases. See CFE_SB_ENBL_RTE1_EID or CFE_SB_ENBL_RTE3_EID

Criticality

This command is not inherently dangerous.

See also

CFE_SB_SEND_ROUTING_INFO_CC, CFE_SB_DISABLE_ROUTE_CC, CFE_SB_RouteCmd_t

Definition at line 234 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.4 CFE_SB_ENABLE_SUB_REPORTING_CC

```
#define CFE_SB_ENABLE_SUB_REPORTING_CC 9
```

Name Enable Subscription Reporting Command

This command will enable subscription reporting and is intended to

be used only by the CFS SBN (Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When subscription reporting is enabled, SB will generate and send a software bus packet for each subscription received. The software bus packet that is sent contains the information received in the subscription API. This subscription report is neeeded by SBN if offboard routing is required.

Command Mnemonic(s) **`$sc_$cpu_SB_EnaSubRptg`**

Command Structure

CFE_SB_CmdHdr_t

Command Verification

Successful execution of this command will result in the sending of a packet (with the CFE_SB_ONESUB_TLM_MID MsgId) for each subscription received by SB through the subscription APIs.

Error Conditions

None

Criticality

None

See also

CFE_SB_SingleSubscriptionTlm_t, CFE_SB_DISABLE_SUB_REPORTING_CC, CFE_SB_SEND_PREV_SUBS_CC

Definition at line 398 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.5 CFE_SB_NOOP_CC

```
#define CFE_SB_NOOP_CC 0
```

Name Software Bus No-Op

Description

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Software Bus task.

Command Mnemonic(s) **`$sc_$cpu_SB_NOOP`**

Command Structure

CFE_SB_CmdHdr_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment
- The CFE_SB_CMD0_RCVD_EID informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Software Bus receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

None

See also

Definition at line 78 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.6 CFE_SB_RESET_COUNTERS_CC

```
#define CFE_SB_RESET_COUNTERS_CC 1
```

Name Software Bus Reset Counters

Description

This command resets the following counters within the Software Bus housekeeping telemetry:

- Command Execution Counter ($sc_$cpu_SB_CMDPC)
- Command Error Counter ($sc_$cpu_SB_CMDEC)

Command Mnemonic(s) **`$sc_$cpu_SB_ResetCtrs`**

Command Structure

CFE_SB_CmdHdr_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment
- The CFE_SB_CMD1_RCVD_EID informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Software Bus receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

See also

Definition at line 115 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.7 CFE_SB_SEND_MAP_INFO_CC

```
#define CFE_SB_SEND_MAP_INFO_CC 8
```

Name Write Map Info to a File

This command will create a file containing the software bus message

map information. The message map is a lookup table (an array of uint16s)that allows fast access to the correct routing table element during a softeware bus send operation. This is diasgnostic information that may be needed due to the dynamic nature of the cFE software bus. An abosulte path and filename may be specified in the command. If this command field contains an empty string (NULL terminator as the first character) the default file path and name is used. The default file path and name is defined in the platform configuration file as CFE_PLATFOR↩ M_SB_DEFAULT_MAP_FILENAME.

Command Mnemonic(s) **`$sc_$cpu_SB_WriteMap2File`**

Command Structure

CFE_SB_WriteFileInfoCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment.
- Specified filename created at specified location. See description.
- The CFE_SB_SND_RTG_EID debug event message will be generated. All debug events are filtered by default.

Error Conditions

- Errors may occur during write operations to the file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_SB_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases. See CFE_SB_SND_RTG_ERR1_EID and CFE_SB_FILEWRITE_ERR_EID

Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_SB_SEND_ROUTING_INFO_CC, CFE_SB_SEND_PIPE_INFO_CC

Definition at line 366 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.8 CFE_SB_SEND_PIPE_INFO_CC

```
#define CFE_SB_SEND_PIPE_INFO_CC 7
```

Name Write Pipe Info to a File

Description

This command will create a file containing the software bus pipe information. The pipe information contains information about every pipe that has been created through the CFE_SB_CreatePipe API. An abosulte path and filename may be specified in the command. If this command field contains an empty string (NULL terminator as the first character) the default file path and name is used. The default file path and name is defined in the platform configuration file as CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME.

Command Mnemonic(s) **`$sc_$cpu_SB_WritePipe2File`**

Command Structure

CFE_SB_WriteFileInfoCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment.
- Specified filename created at specified location. See description.
- The CFE_SB_SND_RTG_EID debug event message will be generated. All debug events are filtered by default.

Error Conditions

- Errors may occur during write operations to the file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_SB_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases. See CFE_SB_SND_RTG_ERR1_EID and CFE_SB_FILEWRITE_ERR_EID

Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_SB_SEND_ROUTING_INFO_CC, CFE_SB_SEND_MAP_INFO_CC

Definition at line 321 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.9 CFE_SB_SEND_PREV_SUBS_CC

```
#define CFE_SB_SEND_PREV_SUBS_CC 11
```

Name Send Previous Subscriptions Command

This command generates a series of packets that contain information

regarding all subscriptions previously received by SB. This command is intended to be used only by the CFS S↩ BN(Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When this command is received the software bus will generate and send a series of packets containing information about all subscription previously received.

Command Mnemonic(s) **`$sc_$cpu_SB_SendPrevSubs`**

Command Structure

CFE_SB_CmdHdr_t

Command Verification

Successful execution of this command will result in a series of packets (with the CFE_SB_ALLSUBS_TLM_MID MsgId) being sent on the software bus.

Error Conditions

None

Criticality

None

See also

CFE_SB_AllSubscriptionsTlm_t, CFE_SB_ENABLE_SUB_REPORTING_CC, CFE_SB_DISABLE_SUB_REP↩ ORTING_CC

Definition at line 462 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.10 CFE_SB_SEND_ROUTING_INFO_CC

```
#define CFE_SB_SEND_ROUTING_INFO_CC 3
```

Name Write Software Bus Routing Info to a File

Description

This command will create a file containing the software bus routing information. The routing information contains information about every subscription that has been received through the SB subscription APIs. An abosulte path and filename may be specified in the command. If this command field contains an empty string (NULL terminator as the first character) the default file path and name is used. The default file path and name is defined in the platform configuration file as CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME.

Command Mnemonic(s) **`$sc_$cpu_SB_WriteRouting2File`**

Command Structure

CFE_SB_WriteFileInfoCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment.
- Specified filename created at specified location. See description.
- The CFE_SB_SND_RTG_EID debug event message will be generated. All debug events are filtered by default.

Error Conditions

- Errors may occur during write operations to the file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_SB_CMDEC`** - command error counter will increment
- A command specific error event message is issued for all error cases. See CFE_SB_SND_RTG_ERR1_EID and CFE_SB_FILEWRITE_ERR_EID

Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_SB_SEND_PIPE_INFO_CC, CFE_SB_SEND_MAP_INFO_CC, CFE_SB_WriteFileInfoCmd_t

Definition at line 194 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.1.11 CFE_SB_SEND_SB_STATS_CC

```
#define CFE_SB_SEND_SB_STATS_CC 2
```

Name Send Software Bus Statistics

Description

This command will cause the SB task to send a statistics packet containing current utilization figures and high water marks which may be useful for checking the margin of the SB platform configuration settings.

Command Mnemonic(s) **`$sc_$cpu_SB_DumpStats`**

Command Structure

CFE_SB_CmdHdr_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_SB_CMDPC`** - command execution counter will increment
- Receipt of statistics packet with MsgId CFE_SB_STATS_TLM_MID
- The CFE_SB_SND_STATS_EID debug event message will be generated. All debug events are filtered by default.

Error Conditions

There are no error conditions for this command. If the Software Bus receives the command, the debug event is sent and the counter is incremented unconditionally.

Criticality

This command is not inherently dangerous. It will create and send a message on the software bus. If performed repeatedly, it is possible that receiver pipes may overflow.

See also

Definition at line 150 of file cfe_sb_msg.h.

Referenced by CFE_SB_ProcessCmdPipePkt().

13.63.2 Typedef Documentation

13.63.2.1 CFE_SB_DisableRoute_t

```
typedef CFE_SB_RouteCmd_t CFE_SB_DisableRoute_t
```

Definition at line 532 of file cfe_sb_msg.h.

13.63.2.2 CFE_SB_DisableSubReporting_t

```
typedef CFE_SB_CmdHdr_t CFE_SB_DisableSubReporting_t
```

Definition at line 479 of file cfe_sb_msg.h.

13.63.2.3 CFE_SB_EnableRoute_t

```
typedef CFE_SB_RouteCmd_t CFE_SB_EnableRoute_t
```

Definition at line 531 of file cfe_sb_msg.h.

13.63.2.4 CFE_SB_EnableSubReporting_t

```
typedef CFE_SB_CmdHdr_t CFE_SB_EnableSubReporting_t
```

Definition at line 478 of file cfe_sb_msg.h.

13.63.2.5 CFE_SB_HKMsg_t

```
typedef CFE_SB_HousekeepingTlm_t CFE_SB_HKMsg_t
```

Definition at line 762 of file cfe_sb_msg.h.

13.63.2.6 CFE_SB_Noop_t

```
typedef CFE_SB_CmdHdr_t CFE_SB_Noop_t
```

Definition at line 476 of file cfe_sb_msg.h.

13.63.2.7 CFE_SB_PrevSubMsg_t

```
typedef CFE_SB_AllSubscriptionsTlm_t CFE_SB_PrevSubMsg_t
```

Definition at line 764 of file cfe_sb_msg.h.

13.63.2.8 CFE_SB_ResetCounters_t

```
typedef CFE_SB_CmdHdr_t CFE_SB_ResetCounters_t
```

Definition at line 477 of file cfe_sb_msg.h.

13.63.2.9 CFE_SB_SendMapInfo_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_SendMapInfo_t
```

Definition at line 506 of file cfe_sb_msg.h.

13.63.2.10 CFE_SB_SendPipeInfo_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_SendPipeInfo_t
```

Definition at line 505 of file cfe_sb_msg.h.

13.63.2.11 CFE_SB_SendPrevSubs_t

```
typedef CFE_SB_CmdHdr_t CFE_SB_SendPrevSubs_t
```

Definition at line 481 of file cfe_sb_msg.h.

13.63.2.12 CFE_SB_SendRoutingInfo_t

```
typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_SendRoutingInfo_t
```

Definition at line 504 of file cfe_sb_msg.h.

13.63.2.13 CFE_SB_SendSbStats_t

```
typedef CFE_SB_CmdHdr_t CFE_SB_SendSbStats_t
```

Definition at line 480 of file cfe_sb_msg.h.

13.63.2.14 CFE_SB_StatMsg_t

```
typedef CFE_SB_StatsTlm_t CFE_SB_StatMsg_t
```

Definition at line 763 of file cfe_sb_msg.h.

13.63.2.15 CFE_SB_SubRprtMsg_t

```
typedef CFE_SB_SingleSubscriptionTlm_t CFE_SB_SubRprtMsg_t
```

Definition at line 765 of file cfe_sb_msg.h.

13.64 cfe/fsw/cfe-core/src/inc/cfe_tbl.h File Reference

```
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_sb_extern_typedefs.h"
#include "common_types.h"
#include "cfe_time.h"
#include "osconfig.h"
```

Data Structures

- struct CFE_TBL_Info_t

Macros

- #define CFE_TBL_OPT_BUFFER_MSK (0x0001)
- #define CFE_TBL_OPT_SNGL_BUFFER (0x0000)
- #define CFE_TBL_OPT_DBL_BUFFER (0x0001)
- #define CFE_TBL_OPT_LD_DMP_MSK (0x0002)
- #define CFE_TBL_OPT_LOAD_DUMP (0x0000)
- #define CFE_TBL_OPT_DUMP_ONLY (0x0002)
- #define CFE_TBL_OPT_USR_DEF_MSK (0x0004)
- #define CFE_TBL_OPT_NOT_USR_DEF (0x0000)
- #define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)

  NOTE: Automatically includes CFE_TBL_OPT_DUMP_ONLY option.
- #define CFE_TBL_OPT_CRITICAL_MSK (0x0008)
- #define CFE_TBL_OPT_NOT_CRITICAL (0x0000)
- #define CFE_TBL_OPT_CRITICAL (0x0008)
- #define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)
- #define CFE_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_FULL_NAME_LEN)
- #define CFE_TBL_BAD_TABLE_HANDLE (CFE_TBL_Handle_t) 0xFFFF
- #define CFE_TBL_INACTIVE_BUFFER CFE_TBL_BufferSelect_INACTIVE
- #define CFE_TBL_ACTIVE_BUFFER CFE_TBL_BufferSelect_ACTIVE

Typedefs

- typedef int32(∗ CFE_TBL_CallbackFuncPtr_t) (void ∗TblPtr)
- typedef int16 CFE_TBL_Handle_t

Enumerations

- enum CFE_TBL_SrcEnum_t { CFE_TBL_SRC_FILE = 0, CFE_TBL_SRC_ADDRESS }

Functions

- int32 CFE_TBL_Register (CFE_TBL_Handle_t ∗TblHandlePtr, const char ∗Name, uint32 Size, uint16 TblOption↩ Flags, CFE_TBL_CallbackFuncPtr_t TblValidationFuncPtr)

  Register a table with cFE to obtain Table Management Services.
- int32 CFE_TBL_Share (CFE_TBL_Handle_t ∗TblHandlePtr, const char ∗TblName)

  Obtain handle of table registered by another application.
- int32 CFE_TBL_Unregister (CFE_TBL_Handle_t TblHandle)

  Unregister a previously registered table and free associated resources.
- int32 CFE_TBL_Load (CFE_TBL_Handle_t TblHandle, CFE_TBL_SrcEnum_t SrcType, const void ∗SrcDataPtr)

  Load a specified table with data from specified source.
- int32 CFE_TBL_Update (CFE_TBL_Handle_t TblHandle)

  Update contents of a specified table, if an update is pending.
- int32 CFE_TBL_GetAddress (void ∗∗TblPtr, CFE_TBL_Handle_t TblHandle)

  Obtain the current address of the contents of the specified table.
- int32 CFE_TBL_ReleaseAddress (CFE_TBL_Handle_t TblHandle)

  Release previously obtained pointer to the contents of the specified table.
- int32 CFE_TBL_GetAddresses (void ∗∗TblPtrs[ ], uint16 NumTables, const CFE_TBL_Handle_t TblHandles[ ])

  Obtain the current addresses of an array of specified tables.
- int32 CFE_TBL_ReleaseAddresses (uint16 NumTables, const CFE_TBL_Handle_t TblHandles[ ])

  Release the addresses of an array of specified tables.
- int32 CFE_TBL_Validate (CFE_TBL_Handle_t TblHandle)

  Perform steps to validate the contents of a table image.
- int32 CFE_TBL_Manage (CFE_TBL_Handle_t TblHandle)

  Perform standard operations to maintain a table.
- int32 CFE_TBL_GetStatus (CFE_TBL_Handle_t TblHandle)

  Obtain current status of pending actions for a table.
- int32 CFE_TBL_GetInfo (CFE_TBL_Info_t ∗TblInfoPtr, const char ∗TblName)

  Obtain characteristics/information of/about a specified table.
- int32 CFE_TBL_DumpToBuffer (CFE_TBL_Handle_t TblHandle)

  Copies the contents of a Dump Only Table to a shared buffer.
- int32 CFE_TBL_Modified (CFE_TBL_Handle_t TblHandle)

  Notify cFE Table Services that table contents have been modified by the Application.
- int32 CFE_TBL_NotifyByMessage (CFE_TBL_Handle_t TblHandle, CFE_SB_MsgId_t MsgId, uint16 CommandCode, uint32 Parameter)

  Instruct cFE Table Services to notify Application via message when table requires management.

13.64.1 Macro Definition Documentation

13.64.1.1 CFE_TBL_ACTIVE_BUFFER

```
#define CFE_TBL_ACTIVE_BUFFER CFE_TBL_BufferSelect_ACTIVE
```

Definition at line 88 of file cfe_tbl.h.

13.64.1.2 CFE_TBL_BAD_TABLE_HANDLE

```
#define CFE_TBL_BAD_TABLE_HANDLE (CFE_TBL_Handle_t) 0xFFFF
```

Definition at line 74 of file cfe_tbl.h.

13.64.1.3 CFE_TBL_INACTIVE_BUFFER

```
#define CFE_TBL_INACTIVE_BUFFER CFE_TBL_BufferSelect_INACTIVE
```

Definition at line 87 of file cfe_tbl.h.

13.64.1.4 CFE_TBL_MAX_FULL_NAME_LEN

```
#define CFE_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_FULL_NAME_LEN)
```

Definition at line 72 of file cfe_tbl.h.

13.64.1.5 CFE_TBL_OPT_BUFFER_MSK

```
#define CFE_TBL_OPT_BUFFER_MSK (0x0001)
```

Definition at line 49 of file cfe_tbl.h.

13.64.1.6 CFE_TBL_OPT_CRITICAL

```
#define CFE_TBL_OPT_CRITICAL (0x0008)
```

Definition at line 63 of file cfe_tbl.h.

13.64.1.7 CFE_TBL_OPT_CRITICAL_MSK

```
#define CFE_TBL_OPT_CRITICAL_MSK (0x0008)
```

Definition at line 61 of file cfe_tbl.h.

13.64.1.8 CFE_TBL_OPT_DBL_BUFFER

```
#define CFE_TBL_OPT_DBL_BUFFER (0x0001)
```

Definition at line 51 of file cfe_tbl.h.

13.64.1.9 CFE_TBL_OPT_DEFAULT

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

Definition at line 65 of file cfe_tbl.h.

13.64.1.10 CFE_TBL_OPT_DUMP_ONLY

```
#define CFE_TBL_OPT_DUMP_ONLY (0x0002)
```

Definition at line 55 of file cfe_tbl.h.

13.64.1.11 CFE_TBL_OPT_LD_DMP_MSK

```
#define CFE_TBL_OPT_LD_DMP_MSK (0x0002)
```

Definition at line 53 of file cfe_tbl.h.

13.64.1.12 CFE_TBL_OPT_LOAD_DUMP

```
#define CFE_TBL_OPT_LOAD_DUMP (0x0000)
```

Definition at line 54 of file cfe_tbl.h.

13.64.1.13 CFE_TBL_OPT_NOT_CRITICAL

```
#define CFE_TBL_OPT_NOT_CRITICAL (0x0000)
```

Definition at line 62 of file cfe_tbl.h.

13.64.1.14 CFE_TBL_OPT_NOT_USR_DEF

```
#define CFE_TBL_OPT_NOT_USR_DEF (0x0000)
```

Definition at line 58 of file cfe_tbl.h.

13.64.1.15 CFE_TBL_OPT_SNGL_BUFFER

```
#define CFE_TBL_OPT_SNGL_BUFFER (0x0000)
```

Definition at line 50 of file cfe_tbl.h.

13.64.1.16 CFE_TBL_OPT_USR_DEF_ADDR

```
#define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)
```

Definition at line 59 of file cfe_tbl.h.

13.64.1.17 CFE_TBL_OPT_USR_DEF_MSK

```
#define CFE_TBL_OPT_USR_DEF_MSK (0x0004)
```

Definition at line 57 of file cfe_tbl.h.

13.64.2 Typedef Documentation

13.64.2.1 CFE_TBL_CallbackFuncPtr_t

```
typedef int32(* CFE_TBL_CallbackFuncPtr_t) (void *TblPtr)
```

Definition at line 97 of file cfe_tbl.h.

13.64.2.2 CFE_TBL_Handle_t

```
typedef int16 CFE_TBL_Handle_t
```

Definition at line 99 of file cfe_tbl.h.

13.64.3 Enumeration Type Documentation

13.64.3.1 CFE_TBL_SrcEnum_t

```
enum CFE_TBL_SrcEnum_t
```

Enumerator

CFE_TBL_SRC_FILE	When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a null terminated character string. The string should specify the full path and filename of the file containing the initial data contents of the table.
CFE_TBL_SRC_ADDRESS	When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a memory location that is the beginning of the initialization data for loading the table OR, in the case of a "user defined" dump only table, the address of the active table itself. The block of memory is assumed to be of the same size specified in the CFE_TBL_Register function Size parameter.

Definition at line 101 of file cfe_tbl.h.

13.64.4 Function Documentation

13.64.4.1 CFE_TBL_DumpToBuffer()

```
int32 CFE_TBL_DumpToBuffer (
            CFE_TBL_Handle_t TblHandle )
```

Description

Copies contents of a Dump Only table to a shared buffer so that it can be written to a file by the Table Services routine. This function is called by the Application that owns the table in response to a CFE_TBL_INFO_DUMP_PENDING status obtained via CFE_TBL_GetStatus.

Assumptions, External Events, and Notes:

1. If the table does not have a dump pending status, nothing will occur (no error, no dump)
2. Applications may wish to use this function in lieu of CFE_TBL_Manage for their Dump Only tables

Parameters

in	*TblHandle*	Handle of Table to be dumped.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Returns

See also

CFE_TBL_Manage

13.64.4.2 CFE_TBL_GetAddress()

```
int32 CFE_TBL_GetAddress (
            void ** TblPtr,
            CFE_TBL_Handle_t TblHandle )
```

Description

When a table has been created and initialized, it is available to any application that can identify it with its unique handle. In order to view the data contained in the table, an application must call this function or CFE_TBL_GetAddresses.

Assumptions, External Events, and Notes:

1. This call can be a blocking call when the table is not double buffered and is shared with another application of lower priority that just happens to be in the middle of a table update of the specific table. If this occurs, the application performing the table update will automatically have its priority elevated in order to release the resource as soon as possible.
2. An application must always release the returned table address using the CFE_TBL_ReleaseAddress or CFE_TBL_ReleaseAddresses function prior to either a CFE_TBL_Update call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

Parameters

in	*TblPtr*	The address of a pointer that will be loaded with the address of the first byte of the table. This pointer can then be typecast by the calling application to the appropriate table data structure.
in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table whose address is to be returned.
out	∗*TblPtr*	Address of the first byte of data associated with the specified table.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_UPDATED	The calling Application has identified a table that has been updated. **NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_UNREGISTERED	The calling application is trying to access a table that has been unregistered.
CFE_TBL_ERR_NEVER_LOADED	This is an error indicating that the table has never been loaded from either a file or a copy from a block of memory so the contents that the returned pointer is pointing to are zeros. **NOTE: Unlike other most other errors, this error condition still returns a valid table pointer. This pointer must be released with the CFE_TBL_ReleaseAddress API before the table can be loaded with data.**

Returns

See also

CFE_TBL_ReleaseAddress, CFE_TBL_GetAddresses, CFE_TBL_ReleaseAddresses

13.64.4.3 CFE_TBL_GetAddresses()

```
int32 CFE_TBL_GetAddresses (
            void ** TblPtrs[],
            uint16 NumTables,
            const CFE_TBL_Handle_t TblHandles[] )
```

Description

When a table has been created and initialized, it is available to any application that can identify it with its unique handle. In order to view the data contained in the table, an application must call this function or CFE_TBL_GetAddresses.

Assumptions, External Events, and Notes:

1. This call can be a blocking call when the table is not double buffered and is shared with another application of lower priority that just happens to be in the middle of a table update of the specific table. If this occurs, the application performing the table update will automatically have its priority elevated in order to release the resource as soon as possible.
2. An application must always release the returned table address using the CFE_TBL_ReleaseAddress or CFE_TBL_ReleaseAddresses function prior to either a CFE_TBL_Update call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

Parameters

in	*TblPtrs*	Array of Pointers to variables that calling Application wishes to hold the start addresses of the Tables.
in	*NumTables*	Size of TblPtrs and TblHandles arrays.
in	*TblHandles*	Array of Table Handles, previously obtained from CFE_TBL_Register or CFE_TBL_Share, of those tables whose start addresses are to be obtained.
out	∗*TblPtrs*	Array of addresses of the first byte of data associated with the specified tables.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_UPDATED	The calling Application has identified a table that has been updated. **NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_UNREGISTERED	The calling application is trying to access a table that has been unregistered.
CFE_TBL_ERR_NEVER_LOADED	This is an error indicating that the table has never been loaded from either a file or a copy from a block of memory so the contents that the returned pointer is pointing to are zeros. **NOTE: Unlike other most other errors, this error condition still returns a valid table pointer. This pointer must be released with the CFE_TBL_ReleaseAddress API before the table can be loaded with data.**

Returns

See also

CFE_TBL_GetAddress, CFE_TBL_ReleaseAddress, CFE_TBL_ReleaseAddresses

13.64.4.4 CFE_TBL_GetInfo()

```
int32 CFE_TBL_GetInfo (
            CFE_TBL_Info_t * TblInfoPtr,
            const char * TblName )
```

Description

This API provides the registry information associated with the specified table. The function fills the given data structure with the data found in the Table Registry.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblInfoPtr*	A pointer to a CFE_TBL_Info_t data structure that is to be populated with table characteristics and information.
in	*TblName*	The processor specific name of the table. It is important to note that the processor specific table name is different from the table name specified in the CFE_TBL_Register API call. The processor specific table name includes the name of the application that created the table. The name would be of the form "ApplicationName.TableName". An example of this would be "ACS.TamParams" for a table called "TamParams" that was registered by the application called "ACS".
out	∗*TblInfoPtr*	Description of the tables characteristics and registry information stored in the CFE_TBL_Info_t data structure format.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_ERR_INVALID_NAME	The calling Application attempted to register a table whose name length exceeded the platform configuration value of CFE_MISSION_TBL_MA↩ X_NAME_LENGTH or was zero characters long.

Returns

See also

CFE_TBL_GetStatus

13.64.4.5 CFE_TBL_GetStatus()

```
int32 CFE_TBL_GetStatus (
            CFE_TBL_Handle_t TblHandle )
```

Description

An application is **required** to perform a periodic check for an update or a validation request for all the tables that it creates. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle. If a table update or validation request is pending, the Application should follow up with a call to CFE_TBL_Update or CFE_TBL_Validate respectively.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be managed.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_UPDATE_PENDING	The calling Application has identified a table that has a load pending.
CFE_TBL_INFO_VALIDATION_PENDING	The calling Application should call CFE_TBL_Validate for the specified table.
CFE_TBL_INFO_DUMP_PENDING	The calling Application should call CFE_TBL_Manage for the specified table. The ground has requested a dump of the Dump-Only table and needs to synchronize with the owning application.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.

CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Returns

See also

CFE_TBL_Manage, CFE_TBL_Update, CFE_TBL_Validate, CFE_TBL_GetInfo

13.64.4.6 CFE_TBL_Load()

```
int32 CFE_TBL_Load (
            CFE_TBL_Handle_t TblHandle,
            CFE_TBL_SrcEnum_t SrcType,
            const void * SrcDataPtr )
```

Description

Once an application has created a table (CFE_TBL_Register), it must provide the values that initialize the contents of that table. The application accomplishes this with one of two different TBL API calls. This function call initializes the table with values that are held in a data structure.

Assumptions, External Events, and Notes:

This function call can block. Therefore, interrupt service routines should NOT initialize their own tables. An application should initialize any table(s) prior to providing the handle(s) to the interrupt service routine.

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be loaded.
in	*SrcType*	Flag indicating the nature of the given `SrcDataPtr` below. This value can be any one of the following: • CFE_TBL_SRC_FILE - When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a null terminated character string. The string should specify the full path and filename of the file containing the initial data contents of the table. • CFE_TBL_SRC_ADDRESS - When this option is selected, the `SrcDataPtr` will be interpreted as a pointer to a memory location that is the beginning of the initialization data for loading the table OR, in the case of a "user defined" dump only table, the address of the active table itself. The block of memory is assumed to be of the same size specified in the CFE_TBL_Register function Size parameter.
in	*SrcDataPtr*	Pointer to either a character string specifying a filename or a memory address of a block of binary data to be loaded into a table or, if the table was registered with the CFE_TBL_OPT_USR_DEF_ADDR option, the address of the active table buffer.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_WARN_SHORT_FILE	The calling Application called CFE_TBL_Load with a filename that specified a file that started with the first byte of the table but contained less data than the size of the table. It should be noted that CFE_TBL_WARN_PARTIAL_LOAD also indicates a partial load (one that starts at a non-zero offset).
CFE_TBL_WARN_PARTIAL_LOAD	The calling Application tried to load a table file whose header claimed the load did not start with the first byteIt should be noted that CFE_TBL_WARN_SHORT_FILE also indicates a partial load.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_DUMP_ONLY	The calling Application has attempted to perform a load on a table that was created with "Dump Only" attributes.
CFE_TBL_ERR_ILLEGAL_SRC_TYPE	The calling Application called CFE_TBL_Load with an illegal value for the second parameter.
CFE_TBL_ERR_LOAD_IN_PROGRESS	The calling Application called CFE_TBL_Load when another Application was trying to load the table.
CFE_TBL_ERR_NO_BUFFER_AVAIL	The calling Application has tried to allocate a working buffer but none were available.
CFE_TBL_ERR_FILE_NOT_FOUND	The calling Application called CFE_TBL_Load with a bad filename.
CFE_TBL_ERR_FILE_TOO_LARGE	The calling Application called CFE_TBL_Load with a filename that specified a file that contained more data than the size of the table OR which contained more data than specified in the table header.
CFE_TBL_ERR_BAD_CONTENT_ID	The calling Application called CFE_TBL_Load with a filename that specified a file whose content ID was not that of a table image.
CFE_TBL_ERR_PARTIAL_LOAD	The calling Application tried to load a table file whose header claimed the load did not start with the first byte and the table image had NEVER been loaded before. Partial loads are not allowed on uninitialized tables. It should be noted that CFE_TBL_WARN_SHORT_FILE also indicates a partial load.

Returns

See also

CFE_TBL_Update, CFE_TBL_Validate, CFE_TBL_Manage

13.64.4.7 CFE_TBL_Manage()

```
int32 CFE_TBL_Manage (
            CFE_TBL_Handle_t TblHandle )
```

Description

An application is **required** to perform a periodic check for an update or a validation request for all the tables that it creates. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle. If a table update or validation request is pending, this function would perform either or both before returning.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be managed.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_UPDATED	The calling Application has identified a table that has been updated. **NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_↩ RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Returns

See also

CFE_TBL_Update, CFE_TBL_Validate, CFE_TBL_Load, CFE_TBL_DumpToBuffer

13.64.4.8 CFE_TBL_Modified()

```
int32 CFE_TBL_Modified (
            CFE_TBL_Handle_t TblHandle )
```

Description

This API notifies Table Services that the contents of the specified table has been modified by the Application. This notification is important when a table has been registered as "Critical" because Table Services can then update the contents of the table kept in the Critical Data Store.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandle*	Handle of Table that was modified.

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_↩ RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Returns

See also

CFE_TBL_Manage

13.64.4.9 CFE_TBL_NotifyByMessage()

```
int32 CFE_TBL_NotifyByMessage (
            CFE_TBL_Handle_t TblHandle,
```

```
CFE_SB_MsgId_t MsgId,
uint16 CommandCode,
uint32 Parameter )
```

Description

This API instructs Table Services to send a message to the calling Application whenever the specified table requires management by the application. This feature allows applications to avoid polling table services via the CFE_TBL_Manage call to determine whether a table requires updates, validation, etc. This API should be called following the CFE_TBL_Register API whenever the owning application requires this feature.

Assumptions, External Events, and Notes:

- Only the application that owns the table is allowed to register a notification message

Parameters

in	*TblHandle*	Handle of Table with which the message should be associated.
in	*MsgId*	Message ID to be used in notification message sent by Table Services.
in	*CommandCode*	Command Code value to be placed in secondary header of message sent by Table Services.
in	*Parameter*	Application defined value to be passed as a parameter in the message sent by Table Services. Suggested use includes an application's table index that allows the same MsgId and Command Code to be used for all table management notifications.

Returns

CFE_SUCCESS	Operation was performed successfully
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

See also

CFE_TBL_Register

13.64.4.10 CFE_TBL_Register()

```
int32 CFE_TBL_Register (
            CFE_TBL_Handle_t * TblHandlePtr,
            const char * Name,
            uint32 Size,
            uint16 TblOptionFlags,
            CFE_TBL_CallbackFuncPtr_t TblValidationFuncPtr )
```

Description

When an application is created and initialized, it is responsible for creating its table images via the TBL API. The application must inform the Table Service of the table name, table size and selection of optional table features.

Assumptions, External Events, and Notes:

Note: This function call can block. Therefore, interrupt service routines should NOT create their own tables. An application should create any table(s) and provide the handle(s) to the interrupt service routine.

Parameters

in	*TblHandlePtr*	a pointer to a CFE_TBL_Handle_t type variable that will be assigned the table's handle. The table handle is required for other API calls when accessing the data contained in the table.
in	*Name*	The application-specific name. This name will be combined with the name of the application to produce a processor specific name of the form "ApplicationName.TableName". The processor specific name will be used in commands for modifying or viewing the contents of the table.
in	*Size*	The size, in bytes, of the table to be created. This is the size that will be allocated as a shared memory resource between the Table Management Service and the calling application.

Parameters

in	*TblOptionFlags*	Flag bits indicating selected options for table. A bitwise OR of the following option flags: • CFE_TBL_OPT_DEFAULT - The default setting for table options is a combination of CFE_TBL_OPT_SNGL_BUFFER and CFE_TBL_OPT_LOAD_DUMP. See below for a description of these two options. This option is mutually exclusive with the CFE_TBL_OPT_DBL_BUFFER, CFE_TBL_OPT_DUMP_ONLY and CFE_TBL_OPT_USR_DEF_ADDR options. • CFE_TBL_OPT_SNGL_BUFFER - When this option is selected, the table will use a shared session table for performing table modifications and a memory copy from the session table to the "active" table buffer will occur when the table is updated. This is the preferred option since it will minimize memory usage. This option is mutually exclusive with the CFE_TBL_OPT_DBL_BUFFER option • CFE_TBL_OPT_DBL_BUFFER - When this option is selected, two instances of the table are created. One is considered the "active" table and the other the "inactive" table. Whenever table modifications occur, they do not require the use of a common session table. Modifications occur in the "inactive" buffer. Then, when it is time to update the table, the pointer to the "active" table is changed to point to the "inactive" buffer thus making it the new "active" buffer. This feature is most useful for time critical applications (ie - interrupt service routines, etc). This option is mutually exclusive with the CFE_TBL_OPT_SNGL_BUFFER and CFE_TBL_OPT_DEFAULT option. • CFE_TBL_OPT_LOAD_DUMP - When this option is selected, the Table Service is allowed to perform all operations on the specified table. This option is mutually exclusive with the CFE_TBL_OPT_DUMP_ONLY option. • CFE_TBL_OPT_DUMP_ONLY - When this option is selected, the Table Service will not perform table loads to this table. This does not prevent, however, a task from writing to the table via an address obtained with the CFE_TBL_GetAddress API function. This option is mutually exclusive with the CFE_TBL_OPT_LOAD_DUMP and CFE_TBL_OPT_DEFAULT options. If the Application wishes to specify their own block of memory as the Dump Only table, they need to also include the CFE_TBL_OPT_USR_DEF_ADDR option explained below. • CFE_TBL_OPT_NOT_USR_DEF - When this option is selected, Table Services allocates memory for the table and, in the case of a double buffered table, it allocates the same amount of memory again for the second buffer. This option is mutually exclusive with the CFE_TBL_OPT_USR_DEF_ADDR option. • CFE_TBL_OPT_USR_DEF_ADDR- When this option is selected, the Table Service will not allocate memory for the table. Table Services will require the Application to identify the location of the active table buffer via the CFE_TBL_Load function. This option implies the CFE_TBL_OPT_DUMP_ONLY and the CFE_TBL_OPT_SNGL_BUFFER options and is mutually exclusive of the CFE_TBL_OPT_DBL_BUFFER option. • CFE_TBL_OPT_CRITICAL- When this option is selected, the Table Service will automatically allocate space in the Critical Data Store (CDS) for the table and insure that the contents in the CDS are the same as the contents of the currently active buffer for the table. This option is mutually exclusive of the CFE_TBL_OPT_USR_DEF_ADDR and CFE_TBL_OPT_DUMP_ONLY options. It should also be noted that the use of this option with double buffered tables will prevent the update of the double buffered table from being

Parameters

in	*TblValidationFuncPtr*	is a pointer to a function that will be executed in the context of the Table Management Service when the contents of a table need to be validated. If set to NULL, then the Table Management Service will assume any data is valid. If the value is not NULL, it must be a pointer to a function with the following prototype: **int32 CallbackFunc(void *TblPtr);** where **TblPtr** will be a pointer to the table data that is to be verified. When the function returns CFE_SUCCESS, the data is considered valid and ready for a commit. When the function returns a negative value, the data is considered invalid and an Event Message will be issued containing the returned value. If the function should return a positive number, the table is considered invalid and the return code is considered invalid. Validation functions **must** return either CFE_SUCCESS or a negative number (whose value is at the developer's discretion). The validation function will be executed in the Application's context so that Event Messages describing the validation failure are possible from within the function.
out	*TblHandlePtr	Handle used to identify table to cFE when performing Table operations. This value is returned at the address specified by TblHandlePtr.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_RECOVERED_TBL	The calling Application registered a critical table whose previous contents were discovered in the Critical Data Store. The discovered contents were copied back into the newly registered table as the table's initial contents. **NOTE:** In this situation, the contents of the table are **NOT** validated using the table's validation function.
CFE_TBL_ERR_DUPLICATE_DIFF_SIZE	An application attempted to register a table with the same name as a table that is already in the registry. The size of the new table is different from the size already in the registry.
CFE_TBL_ERR_DUPLICATE_NOT_OWNED	An application attempted to register a table with the same name as a table that is already in the registry. The previously registered table is owned by a different application.
CFE_TBL_ERR_REGISTRY_FULL	An application attempted to create a table and the Table registry already contained CFE_PLATFORM_TBL_MAX_NUM_TABLES in it.
CFE_TBL_ERR_HANDLES_FULL	An application attempted to create a table and the Table Handle Array already used all CFE_PLATFORM_TBL_MAX_NUM_HANDLES in it.
CFE_TBL_ERR_INVALID_SIZE	The calling Application attempted to register a table: a) that was a double buffered table with size greater than CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE b) that was a single buffered table with size greater than CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE c) that had a size of zero
CFE_TBL_ERR_INVALID_NAME	The calling Application attempted to register a table whose name length exceeded the platform configuration value of CFE_MISSION_TBL_MAX_NAME_LENGTH or was zero characters long.

CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

CFE_TBL_Unregister, CFE_TBL_Share

13.64.4.11 CFE_TBL_ReleaseAddress()

```
int32 CFE_TBL_ReleaseAddress (
            CFE_TBL_Handle_t TblHandle )
```

Description

Each application is **required** to release a table address obtained through the CFE_TBL_GetAddress function.

Assumptions, External Events, and Notes:

An application must always release the returned table address using the CFE_TBL_ReleaseAddress function prior to either a CFE_TBL_Update call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table whose address is to be released.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_UPDATED	The calling Application has identified a table that has been updated. **NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_↩ RegisterApp function.

CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_NEVER_LOADED	This is an error indicating that the table has never been loaded from either a file or a copy from a block of memory so the contents that the returned pointer is pointing to are zeros. **NOTE: Unlike other most other errors, this error condition still returns a valid table pointer. This pointer must be released with the CFE_TBL_ReleaseAddress API before the table can be loaded with data.**

Returns

See also

CFE_TBL_GetAddress, CFE_TBL_GetAddresses, CFE_TBL_ReleaseAddresses

13.64.4.12 CFE_TBL_ReleaseAddresses()

```
int32 CFE_TBL_ReleaseAddresses (
            uint16 NumTables,
            const CFE_TBL_Handle_t TblHandles[] )
```

Description

Each application is **required** to release a table address obtained through the CFE_TBL_GetAddress function.

Assumptions, External Events, and Notes:

An application must always release the returned table address using the CFE_TBL_ReleaseAddress function prior to either a CFE_TBL_Update call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

Parameters

in	*NumTables*	Size of TblHandles array.
in	*TblHandles*	Array of Table Handles, previously obtained from CFE_TBL_Register or CFE_TBL_Share, of those tables whose start addresses are to be released.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_UPDATED	The calling Application has identified a table that has been updated. **NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_↩ RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_NEVER_LOADED	This is an error indicating that the table has never been loaded from either a file or a copy from a block of memory so the contents that the returned pointer is pointing to are zeros. **NOTE: Unlike other most other errors, this error condition still returns a valid table pointer. This pointer must be released with the CFE_TBL_ReleaseAddress API before the table can be loaded with data.**

Returns

See also

CFE_TBL_GetAddress, CFE_TBL_ReleaseAddress, CFE_TBL_GetAddresses

13.64.4.13 CFE_TBL_Share()

```
int32 CFE_TBL_Share (
            CFE_TBL_Handle_t * TblHandlePtr,
            const char * TblName )
```

Description

After a table has been created, other applications can gain access to that table via the table handle. In order for two or more applications to share a table, the applications that do not create the table must obtain the handle using this function.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandlePtr*	A pointer to a CFE_TBL_Handle_t type variable that will be assigned the table's handle. The table handle is required for other API calls when accessing the data contained in the table.
in	*TblName*	The processor specific name of the table. It is important to note that the processor specific table name is different from the table name specified in the CFE_TBL_Register API call. The processor specific table name includes the name of the application that created the table. The name would be of the form "ApplicationName.TableName". An example of this would be "ACS.TamParams" for a table called "TamParams" that was registered by the application called "ACS".
out	∗*TblHandlePtr*	Handle used to identify table to cFE when performing Table operations. This value is returned at the address specified by TblHandlePtr.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_ERR_HANDLES_FULL	An application attempted to create a table and the Table Handle Array already used all CFE_PLATFORM_TBL_MAX_NUM_HANDLES in it.
CFE_TBL_ERR_INVALID_NAME	The calling Application attempted to register a table whose name length exceeded the platform configuration value of CFE_MISSION_TBL_MAX_NAME_LENGTH or was zero characters long.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

CFE_TBL_Unregister, CFE_TBL_Register

13.64.4.14 CFE_TBL_Unregister()

```
int32 CFE_TBL_Unregister (
            CFE_TBL_Handle_t TblHandle )
```

Description

When an application is being removed from the system, it should unregister those tables that it created. The application should call this function as a part of its cleanup process. The table will be removed from memory once all table addresses referencing it have been released.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be unregistered.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_↩ RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

CFE_TBL_Share, CFE_TBL_Register

13.64.4.15 CFE_TBL_Update()

```
int32 CFE_TBL_Update (
            CFE_TBL_Handle_t TblHandle )
```

Description

An application is **required** to perform a periodic check for an update for all the tables that it creates. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle or at regular intervals. To determine whether an update is pending prior to making this call, the Application can use the CFE_TBL_GetStatus API first. If a table update is pending, it will take place during this function call.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be updated.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_NO_UPDATE_PENDING	The calling Application has attempted to update a table without a pending load.
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)

Returns

See also

CFE_TBL_Load, CFE_TBL_Validate, CFE_TBL_Manage

13.64.4.16 CFE_TBL_Validate()

```
int32 CFE_TBL_Validate (
            CFE_TBL_Handle_t TblHandle )
```

Description

An application is **required** to perform a periodic check for an update or a validation request for all the tables that it creates. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle. To determine whether a validation request is pending prior to making this call, the Application can use the CFE_TBL_GetStatus API first. If a table validation is pending, the Application would call this function to perform the necessary actions.

Assumptions, External Events, and Notes:

None

Parameters

in	*TblHandle*	Handle, previously obtained from CFE_TBL_Register or CFE_TBL_Share, that identifies the Table to be managed.

CFE_SUCCESS	Operation was performed successfully
CFE_TBL_INFO_NO_VALIDATION_PENDING	The calling Application tried to validate a table that did not have a validation request pending.
CFE_ES_ERR_APPNAME	There is no match for the given application name in the current application list.
CFE_ES_ERR_BUFFER	Invalid pointer argument (NULL)
CFE_TBL_ERR_BAD_APP_ID	The calling application does not have a legitimate Application ID. Most likely cause is a failure to register with the cFE via the CFE_ES_RegisterApp function.
CFE_TBL_ERR_NO_ACCESS	The calling application either failed when calling CFE_TBL_Register, failed when calling CFE_TBL_Share or forgot to call either one.
CFE_TBL_ERR_INVALID_HANDLE	The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Returns

See also

CFE_TBL_Update, CFE_TBL_Manage, CFE_TBL_Load

13.65 cfe/fsw/cfe-core/src/inc/cfe_tbl_events.h File Reference

Macros

- #define CFE_TBL_MAX_EID 98

Informational Event Message IDs

- #define CFE_TBL_INIT_INF_EID 1
  'Task Initialized'

Command Response Informational Event Message IDs

- #define CFE_TBL_NOOP_INF_EID 10
  'No-op command'

- #define CFE_TBL_RESET_INF_EID 11
  'Reset Counters command'
- #define CFE_TBL_FILE_LOADED_INF_EID 12
  'Successful load of '%s' into '%s' working buffer'
- #define CFE_TBL_OVERWRITE_DUMP_INF_EID 13
  'Successfully overwrote '%s' with Table '%s''
- #define CFE_TBL_WRITE_DUMP_INF_EID 14
  'Successfully dumped Table '%s' to '%s''
- #define CFE_TBL_OVERWRITE_REG_DUMP_INF_EID 15
  'Successfully overwrote '%s' with Table Registry'
- #define CFE_TBL_VAL_REQ_MADE_INF_EID 16
  'Tbl Services issued validation request for '%s''
- #define CFE_TBL_LOAD_PEND_REQ_INF_EID 17
  'Tbl Services notifying App that '%s' has a load pending'
- #define CFE_TBL_TLM_REG_CMD_INF_EID 18
  'Table Registry entry for '%s' will be telemetered'
- #define CFE_TBL_LOAD_ABORT_INF_EID 21
  'Table Load Aborted for '%s''
- #define CFE_TBL_WRITE_REG_DUMP_INF_EID 22
  'Successfully dumped Table Registry to '%s':Size=%d,Entries=%d'
- #define CFE_TBL_ASSUMED_VALID_INF_EID 23
  'Tbl Services assumes '%s' is valid. No Validation Function has been registered'

Command Error Event Message IDs

- #define CFE_TBL_MID_ERR_EID 50
  'Invalid message ID - ID = 0x%X'
- #define CFE_TBL_CC1_ERR_EID 51
  'Invalid command code - ID = 0x%X, CC = %d'
- #define CFE_TBL_LEN_ERR_EID 52
  'Invalid cmd pkt - ID = 0x%X, CC = %d, Len = %d'
- #define CFE_TBL_FILE_ACCESS_ERR_EID 53
  'Unable to open file '%s' for table load, Status = 0x%08X'
- #define CFE_TBL_FILE_STD_HDR_ERR_EID 54
  'Unable to read std header for '%s', Status = 0x%08X'
- #define CFE_TBL_FILE_TBL_HDR_ERR_EID 55
  'Unable to read tbl header for '%s', Status = 0x%08X'
- #define CFE_TBL_FAIL_HK_SEND_ERR_EID 56
  'Unable to send Hk Packet (Status=0x%08X)'
- #define CFE_TBL_NO_SUCH_TABLE_ERR_EID 57
  'Unable to locate '%s' in Table Registry'
- #define CFE_TBL_FILE_TYPE_ERR_EID 58
  'File '%s' is not a cFE file type, ContentType = 0x%08X'
- #define CFE_TBL_FILE_SUBTYPE_ERR_EID 59
  'File subtype for '%s' is wrong. Subtype = 0x%08X'
- #define CFE_TBL_NO_WORK_BUFFERS_ERR_EID 60
  'No working buffers available for table '%s''
- #define CFE_TBL_INTERNAL_ERROR_ERR_EID 61
  'Internal Error (Status=0x%08X)'
- #define CFE_TBL_CREATING_DUMP_FILE_ERR_EID 62
  'Error creating dump file '%s', Status=0x%08X'
- #define CFE_TBL_WRITE_CFE_HDR_ERR_EID 63
  'Error writing cFE File Header to '%s', Status=0x%08X'
- #define CFE_TBL_WRITE_TBL_HDR_ERR_EID 64

'Error writing Tbl image File Header to '%s', Status=0x%08X'

- #define CFE_TBL_WRITE_TBL_IMG_ERR_EID 65

  'Error writing Tbl image to '%s', Status=0x%08X'
- #define CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID 66

  'No Inactive Buffer for Table '%s' present'
- #define CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID 67

  'Too many Table Validations have been requested'
- #define CFE_TBL_WRITE_TBL_REG_ERR_EID 68

  'Error writing Registry to '%s', Status=0x%08X'
- #define CFE_TBL_LOAD_ABORT_ERR_EID 69

  'Cannot abort load of '%s'. No load started.'
- #define CFE_TBL_ACTIVATE_ERR_EID 70

  'Cannot activate table '%s'. No Inactive image available'
- #define CFE_TBL_FILE_INCOMPLETE_ERR_EID 71

  'Incomplete load of '%s' into '%s' working buffer'
- #define CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID 72

  'Cannot load '%s' (%d) at offset %d in '%s' (%d)'
- #define CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID 73

  'Table Hdr in '%s' indicates no data in file'
- #define CFE_TBL_PARTIAL_LOAD_ERR_EID 74

  ''%s' has partial load for uninitialized table '%s''
- #define CFE_TBL_FILE_TOO_BIG_ERR_EID 75

  'File '%s' has more data than Tbl Hdr indicates (%d)'
- #define CFE_TBL_TOO_MANY_DUMPS_ERR_EID 76

  'Too many Dump Only Table Dumps have been requested'
- #define CFE_TBL_DUMP_PENDING_ERR_EID 77

  'A dump for '%s' is already pending'
- #define CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID 78

  'Illegal attempt to activate dump-only table '%s''
- #define CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID 79

  'Attempted to load DUMP-ONLY table '%s' from '%s''
- #define CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID 80

  'Cmd for Table '%s' had illegal buffer parameter (0x%08X)'
- #define CFE_TBL_UNVALIDATED_ERR_EID 81

  'Cannot activate table '%s'. Inactive image not Validated'
- #define CFE_TBL_IN_REGISTRY_ERR_EID 82

  ''%s' found in Table Registry. CDS cannot be deleted until table is unregistered'
- #define CFE_TBL_NOT_CRITICAL_TBL_ERR_EID 83

  'Table '%s' is in Critical Table Registry but CDS is not tagged as a table'
- #define CFE_TBL_NOT_IN_CRIT_REG_ERR_EID 84

  'Table '%s' is not found in Critical Table Registry'
- #define CFE_TBL_CDS_NOT_FOUND_ERR_EID 85

  'Unable to locate '%s' in CDS Registry'
- #define CFE_TBL_CDS_DELETE_ERR_EID 86

  'Error while deleting '%s' from CDS, See SysLog.(Err=0x%08X)'
- #define CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID 87

  'CDS '%s' owning app is still active'
- #define CFE_TBL_LOADING_PENDING_ERR_EID 88

  'Attempted to load table '%s' while previous load is still pending'
- #define CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID 89

  'Manage Notification Pkt Error(MsgId=0x%08X, CC=0x%04X, Param=0x%08X, Status=0x%08X)'

API Informational Event Message IDs

- #define CFE_TBL_LOAD_SUCCESS_INF_EID 35
  'Successfully loaded '%s' from '%s''
- #define CFE_TBL_VALIDATION_INF_EID 36
  '%s validation successful for Inactive '%s''
- #define CFE_TBL_UPDATE_SUCCESS_INF_EID 37
  '%s Successfully Updated '%s''
- #define CFE_TBL_CDS_DELETED_INFO_EID 38
  'Successfully removed '%s' from CDS'

API Error Event Message IDs

- #define CFE_TBL_REGISTER_ERR_EID 90
  '%s Failed to Register '%s', Status=0x%08X'
- #define CFE_TBL_SHARE_ERR_EID 91
  '%s Failed to Share '%s', Status=0x%08X'
- #define CFE_TBL_UNREGISTER_ERR_EID 92
  '%s Failed to Unregister '%s', Status=0x%08X'
- #define CFE_TBL_LOAD_ERR_EID 93
  '%s Failed to Load '%s' from %s, Status=0x%08X"
- #define CFE_TBL_LOAD_TYPE_ERR_EID 94
  '%s Failed to Load '%s' (Invalid Source Type)"
- #define CFE_TBL_UPDATE_ERR_EID 95
  '%s Failed to Update '%s', Status=0x%08X"
- #define CFE_TBL_VALIDATION_ERR_EID 96
  '%s validation failed for Inactive '%s', Status=0x%08X"
- #define CFE_TBL_SPACECRAFT_ID_ERR_EID 97
  'Unable to verify Spacecraft ID for '%s', ID = 0x%08X'
- #define CFE_TBL_PROCESSOR_ID_ERR_EID 98
  'Unable to verify Processor ID for '%s', ID = 0x%08X'

13.65.1 Macro Definition Documentation

13.65.1.1 CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID

```
#define CFE_TBL_ACTIVATE_DUMP_ONLY_ERR_EID 78
```

Event Message `'Illegal attempt to activate dump-only table '%s''`

Type: ERROR

Cause:

This event message is generated when a Table Activate command for a Dump-Only Table was received. By definition, Dump-Only tables are not allowed to be loaded with any new data.

Definition at line 692 of file cfe_tbl_events.h.

13.65.1.2 CFE_TBL_ACTIVATE_ERR_EID

```
#define CFE_TBL_ACTIVATE_ERR_EID 70
```

Event Message `'Cannot activate table '%s'.  No Inactive image available'`

Type: ERROR

Cause:

This event message is generated when an Activate Table command is received and the command specified table does not currently have an inactive buffer associated with it.

Definition at line 585 of file cfe_tbl_events.h.

13.65.1.3 CFE_TBL_ASSUMED_VALID_INF_EID

```
#define CFE_TBL_ASSUMED_VALID_INF_EID 23
```

Event Message `'Tbl Services assumes '%s' is valid.  No Validation Function has been registered'`

Type: INFORMATION

Cause:

This event message is generated when Table Services has received a Validation Command for a table that never specified a Validation Function when it was registered via the CFE_TBL_Register API.

Definition at line 239 of file cfe_tbl_events.h.

13.65.1.4 CFE_TBL_CC1_ERR_EID

```
#define CFE_TBL_CC1_ERR_EID 51
```

Event Message `'Invalid command code - ID = 0x%X, CC = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_TBL_CMD_MID message ID has arrived but whose Command Code is not one of the command codes specified in cfe_tbl_msg.h . This problem is most likely to occur when:

1. A Message ID meant for another Application became corrupted and was set equal to CFE_TBL_CMD_MID.
2. The Command Code field in the Message became corrupted.
3. The command database at the ground station has been corrupted.

The `ID` field in the event message specifies the Message ID (in hex) and the `CC` field specifies the Command Code (in decimal) found in the message.

Definition at line 283 of file cfe_tbl_events.h.

13.65.1.5 CFE_TBL_CDS_DELETE_ERR_EID

```
#define CFE_TBL_CDS_DELETE_ERR_EID 86
```

Event Message `'Error while deleting '%s' from CDS, See SysLog.(Err=0x%08X)'`

Type: ERROR

Cause:

This event message is generated when an unexpected error was encountered during the deletion of the CDS. The System Log should have more precise information on the nature of the error.

Definition at line 798 of file cfe_tbl_events.h.

13.65.1.6 CFE_TBL_CDS_DELETED_INFO_EID

```
#define CFE_TBL_CDS_DELETED_INFO_EID 38
```

Event Message `'Successfully removed '%s' from CDS'`

Type: INFORMATION

Cause:

This event message is generated when a Critical Table's CDS has been successfully deleted.

Definition at line 894 of file cfe_tbl_events.h.

13.65.1.7 CFE_TBL_CDS_NOT_FOUND_ERR_EID

```
#define CFE_TBL_CDS_NOT_FOUND_ERR_EID 85
```

Event Message `'Unable to locate '%s' in CDS Registry'`

Type: ERROR

Cause:

This event message is generated when a Table Delete Critical Data Store command is received specifying a table name that WAS found in the Critical Table Registry but its associated entry in the Critical Data Store Registry was not found. Somehow the two entities have become out of synch.

Definition at line 786 of file cfe_tbl_events.h.

13.65.1.8 CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID

```
#define CFE_TBL_CDS_OWNER_ACTIVE_ERR_EID 87
```

Event Message `'CDS '%s' owning app is still active'`

Type: ERROR

Cause:

This event message is generated when an attempt is made to delete a CDS while an application with the same name as the CDS Prefix is still registered in the system. Owning applications must not be active before an associated CDS can be deleted.

Definition at line 811 of file cfe_tbl_events.h.

13.65.1.9 CFE_TBL_CREATING_DUMP_FILE_ERR_EID

```
#define CFE_TBL_CREATING_DUMP_FILE_ERR_EID 62
```

Event Message `'Error creating dump file '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when a Table Dump or Table Registry Dump command was received and the cFE Table Services is unable to create the specified file.

The `Status` field provides the return status from the OS_creat function call.

Definition at line 471 of file cfe_tbl_events.h.

13.65.1.10 CFE_TBL_DUMP_PENDING_ERR_EID

```
#define CFE_TBL_DUMP_PENDING_ERR_EID 77
```

Event Message `'A dump for '%s' is already pending'`

Type: ERROR

Cause:

This event message is generated when a Table Dump command for a Dump-Only Table was received and Table Services hasn't finished processing the previous Table Dump command for the same Table.

Definition at line 680 of file cfe_tbl_events.h.

13.65.1.11 CFE_TBL_FAIL_HK_SEND_ERR_EID

```
#define CFE_TBL_FAIL_HK_SEND_ERR_EID 56
```

Event Message `'Unable to send Hk Packet (Status=0x%08X)'`

Type: ERROR

Cause:

This event message is generated when failure occurs while attempting to send the Housekeeping Message over the Software Bus.

The `Status` field of the event message contains the error code returned by CFE_SB_SendMsg.

Definition at line 368 of file cfe_tbl_events.h.

13.65.1.12 CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID

```
#define CFE_TBL_FAIL_NOTIFY_SEND_ERR_EID 89
```

Event Message `'Manage Notification Pkt Error(MsgId=0x%08X, CC=0x%04X, Param=0x%08X, Status=0x%08X)'`

Type: ERROR

Cause:

This event message is generated when a table management notification message fails to be sent via the software bus.

The `MsgId` is the message ID of the table management notification message that was attempted to be sent, the `CC` is the command code, the `Param` is the application specified command parameter and the `Status` is the error code returned by the CFE_SB_SendMsg API call.

Definition at line 841 of file cfe_tbl_events.h.

13.65.1.13 CFE_TBL_FILE_ACCESS_ERR_EID

```
#define CFE_TBL_FILE_ACCESS_ERR_EID 53
```

Event Message `'Unable to open file '%s' for table load, Status = 0x%08X'`

Type: ERROR

Cause:

This event message is generated upon receipt of a Load Table command when the specified file containing the table image to be loaded cannot be opened. Possible causes for this are:

1. The filename was misspelled
2. The path to the file was incorrect
3. The length of the filename and/or path exceeds the allowable length (see OS_MAX_PATH_LEN and OS_MAX_FILE_NAME, respectively)

The `Status` field in the event message indicates the error code returned by the OS_open API.

Definition at line 322 of file cfe_tbl_events.h.

13.65.1.14 CFE_TBL_FILE_INCOMPLETE_ERR_EID

```
#define CFE_TBL_FILE_INCOMPLETE_ERR_EID 71
```

Event Message `'Incomplete load of '%s' into '%s' working buffer'`

Type: ERROR

Cause:

This event message is generated when a Load Table command is received and the Table Services is unable to load the number of bytes specified in the Table Image Header of the command specified file from the file into the Inactive Buffer.

Definition at line 598 of file cfe_tbl_events.h.

13.65.1.15 CFE_TBL_FILE_LOADED_INF_EID

```
#define CFE_TBL_FILE_LOADED_INF_EID 12
```

Event Message `'Successful load of '%s' into '%s' working buffer'`

Type: INFORMATION

Cause:

This event message is always generated after a successful execution of a cFE Table Services Load Table command

Definition at line 107 of file cfe_tbl_events.h.

13.65.1.16 CFE_TBL_FILE_STD_HDR_ERR_EID

```
#define CFE_TBL_FILE_STD_HDR_ERR_EID 54
```

Event Message `'Unable to read std header for '%s', Status = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when a read failure occurs during the reading of the cFE Standard File Header of a table image file specified either by an Application calling the CFE_TBL_Load API or in response to a command to Table Services requesting a table image file be loaded into an inactive buffer.

The `Status` field of the event message contains the error code returned by CFE_FS_ReadHeader.

Definition at line 338 of file cfe_tbl_events.h.

13.65.1.17 CFE_TBL_FILE_SUBTYPE_ERR_EID

```
#define CFE_TBL_FILE_SUBTYPE_ERR_EID 59
```

Event Message `'File subtype for '%s' is wrong.  Subtype = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when either an Application calls the CFE_TBL_Load API or a Table Load command has been received and the specified file has a cFE Standard File Header whose Sub Type is not equal to the expected CFE_FS_SubType_TBL_IMG. Most likely causes for this are:

1. The specified file is not a cFE table image file.
2. The specified file has been created with bad "endianess" (headers should always conform to a big endian format).
3. The specified file has become corrupted.

The `SubType` field specified in the event message contains the sub type that was found in the specified file.

Definition at line 427 of file cfe_tbl_events.h.

13.65.1.18 CFE_TBL_FILE_TBL_HDR_ERR_EID

```
#define CFE_TBL_FILE_TBL_HDR_ERR_EID 55
```

Event Message `'Unable to read tbl header for '%s', Status = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when a read failure occurs during the reading of the cFE Table File Secondary Header of a table image file specified either by an Application calling the CFE_TBL_Load API or in response to a command to Table Services requesting a table image file be loaded into an inactive buffer.

The `Status` field of the event message contains the error code returned by OS_read.

Definition at line 354 of file cfe_tbl_events.h.

13.65.1.19 CFE_TBL_FILE_TOO_BIG_ERR_EID

```
#define CFE_TBL_FILE_TOO_BIG_ERR_EID 75
```

Event Message `'File '%s' has more data than Tbl Hdr indicates (%d)'`

Type: ERROR

Cause:

This event message is generated when a Load Table command is received and Table Services is able to locate more data in the specified Table Image file than the Table Header claims is present.

Definition at line 654 of file cfe_tbl_events.h.

13.65.1.20 CFE_TBL_FILE_TYPE_ERR_EID

```
#define CFE_TBL_FILE_TYPE_ERR_EID 58
```

Event Message `'File '%s' is not a cFE file type, ContentType = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when either an Application calls the CFE_TBL_Load API or a Table Load command has been received and the specified file has a cFE Standard File Header whose Content Type is not equal to the expected CFE_FS_FILE_CONTENT_ID. Most likely causes for this are:

1. The specified file is not a cFE compatible file.
2. The specified file has been created with bad "endianess" (headers should always conform to a big endian format).
3. The specified file has become corrupted.

The `ContentType` field specified in the event message contains the content type that was found in the specified file.

Definition at line 406 of file cfe_tbl_events.h.

13.65.1.21 CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID

```
#define CFE_TBL_ILLEGAL_BUFF_PARAM_ERR_EID 80
```

Event Message `'Cmd for Table '%s' had illegal buffer parameter (0x%08X)'`

Type: ERROR

Cause:

This event message is generated when either a Table Validate command or a Table Dump Command contains a buffer identifier that does not equal either of the valid values (see CFE_TBL_DumpCmd_Payload_t::ActiveTableFlag or CFE_TBL_ValidateCmd_Payload_t::ActiveTableFlag)

The parameter in the Event Message indicates (in hex) the value found for the ActiveTableFlag in the command.

Definition at line 719 of file cfe_tbl_events.h.

13.65.1.22 CFE_TBL_IN_REGISTRY_ERR_EID

```
#define CFE_TBL_IN_REGISTRY_ERR_EID 82
```

Event Message ''%s' found in Table Registry. CDS cannot be deleted until table is unregistered'

Type: ERROR

Cause:

This event message is generated when a Table Delete Critical Data Store command is received specifying a Table Image that is still registered. Critical Table Images cannot be removed from the CDS until the table is first removed from the Registry. Unload the owning application and try again.

Definition at line 746 of file cfe_tbl_events.h.

13.65.1.23 CFE_TBL_INIT_INF_EID

```
#define CFE_TBL_INIT_INF_EID 1
```

Event Message 'Task Initialized'

Type: INFORMATION

Cause:

This event message is always automatically issued when the Table Services Task completes its Initialization.

Definition at line 68 of file cfe_tbl_events.h.

13.65.1.24 CFE_TBL_INTERNAL_ERROR_ERR_EID

```
#define CFE_TBL_INTERNAL_ERROR_ERR_EID 61
```

Event Message `'Internal Error (Status=0x%08X)'`

Type: ERROR

Cause:

This event message is generated when a Table Load command was issued and the cFE Table Services is unable to allocate a working table buffer for an unexpected reason.

The `Status` field provides the return status from the function that was to provide a working buffer.

Definition at line 457 of file cfe_tbl_events.h.

13.65.1.25 CFE_TBL_LEN_ERR_EID

```
#define CFE_TBL_LEN_ERR_EID 52
```

Event Message `'Invalid cmd pkt - ID = 0x%X, CC = %d, Len = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_TBL_CMD_MID message ID has arrived but whose packet length does not match the expected length for the specified command code.

The `ID` field in the event message specifies the Message ID (in hex), the `CC` field specifies the Command Code (in decimal) and `Len` specifies the message Length (in decimal) found in the message.

Definition at line 300 of file cfe_tbl_events.h.

13.65.1.26 CFE_TBL_LOAD_ABORT_ERR_EID

```
#define CFE_TBL_LOAD_ABORT_ERR_EID 69
```

Event Message `'Cannot abort load of '%s'.  No load started.'`

Type: ERROR

Cause:

This event message is generated when an Abort Load command is received and the command specified table is not currently in the process of being loaded.

Definition at line 573 of file cfe_tbl_events.h.

13.65.1.27 CFE_TBL_LOAD_ABORT_INF_EID

```
#define CFE_TBL_LOAD_ABORT_INF_EID 21
```

Event Message `'Table Load Aborted for '%s''`

Type: INFORMATION

Cause:

This event message is generated upon successful execution of a cFE Table Services Abort Table Load command .

Definition at line 211 of file cfe_tbl_events.h.

13.65.1.28 CFE_TBL_LOAD_ERR_EID

```
#define CFE_TBL_LOAD_ERR_EID 93
```

Event Message `'%s Failed to Load '%s' from %s, Status=0x%08X"`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Load unsuccessfully.

The `Status` field of the Event Message can be used to identify the reason for the failure by looking it up in the cfe_error.h file

Definition at line 955 of file cfe_tbl_events.h.

13.65.1.29 CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID

```
#define CFE_TBL_LOAD_EXCEEDS_SIZE_ERR_EID 72
```

Event Message `'Cannot load '%s' (%d) at offset %d in '%s' (%d)'`

Type: ERROR

Cause:

This event message is generated when a Load Table command is received and the Table Header in the specified Table Image file identifies a number of bytes with a specified starting offset that would exceed the size of the specified table. For example, if a table had 10 bytes and the Table Header indicated that the Table Image in the file contains 7 bytes that starts at offset 5, then the data content would have exceeded the 10 byte limit of the table.

The numbers in parenthesis in the event message text indicate the data size (in bytes) for the specified load file and the registered size for the specified table.

Definition at line 616 of file cfe_tbl_events.h.

13.65.1.30 CFE_TBL_LOAD_PEND_REQ_INF_EID

```
#define CFE_TBL_LOAD_PEND_REQ_INF_EID 17
```

Event Message `'Tbl Services notifying App that '%s' has a load pending'`

Type: DEBUG

Cause:

This event message is generated upon successful execution of a cFE Table Services Activate Table command . It should be noted, however, that this Event Message does *NOT* indicate completion of the Table Activation. It is *ONLY* indicating that the appropriate flag has been set to *NOTIFY* the table's owning Application that an Update has been requested. Completion of the Update is indicated by either the CFE_TBL_UPDATE_SUCCESS_INF_EID or CFE_TBL_UPDAT↩ E_ERR_EID event messages.

Definition at line 186 of file cfe_tbl_events.h.

13.65.1.31 CFE_TBL_LOAD_SUCCESS_INF_EID

```
#define CFE_TBL_LOAD_SUCCESS_INF_EID 35
```

Event Message `'Successfully loaded '%s' from '%s''`

Type: DEBUG (the first time) and INFORMATION (normally)

Cause:

This event message is generated when a Table is successfully updated by its owning Application with the contents of the Application specified file or memory area. This Event Message only appears when an Application successfully calls the CFE_TBL_Load API.

Definition at line 858 of file cfe_tbl_events.h.

13.65.1.32 CFE_TBL_LOAD_TYPE_ERR_EID

```
#define CFE_TBL_LOAD_TYPE_ERR_EID 94
```

Event Message `'%s Failed to Load '%s' (Invalid Source Type)"`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Load with a bad value for the `SrcType` parameter. The `SrcType` must be one of the values specified by CFE_TBL_SrcEnum_t.

Definition at line 967 of file cfe_tbl_events.h.

13.65.1.33 CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID

```
#define CFE_TBL_LOADING_A_DUMP_ONLY_ERR_EID 79
```

Event Message `'Attempted to load DUMP-ONLY table '%s' from '%s''`

Type: ERROR

Cause:

This event message is generated when a Table Load command for a Dump-Only Table was received. By definition, Dump-Only tables are not allowed to be loaded with any new data.

Definition at line 704 of file cfe_tbl_events.h.

13.65.1.34 CFE_TBL_LOADING_PENDING_ERR_EID

```
#define CFE_TBL_LOADING_PENDING_ERR_EID 88
```

Event Message `'Attempted to load table '%s' while previous load is still pending'`

Type: ERROR

Cause:

This event message is generated when an attempt is made to load a table while a previous load is still pending. The most likely cause of this is the owning application is waiting for an appropriate time to load the table with the specified contents. In order to override this load, the user would be required to issue the Abort Load Command .

Definition at line 825 of file cfe_tbl_events.h.

13.65.1.35 CFE_TBL_MAX_EID

```
#define CFE_TBL_MAX_EID 98
```

Definition at line 49 of file cfe_tbl_events.h.

13.65.1.36 CFE_TBL_MID_ERR_EID

```
#define CFE_TBL_MID_ERR_EID 50
```

Event Message `'Invalid message ID - ID = 0x%X'`

Type: ERROR

Cause:

This event message is generated when a message has arrived on the cFE Table Services Application's Message Pipe that has a Message ID that is neither CFE_TBL_SEND_HK_MID or CFE_TBL_CMD_MID. Most likely, the cFE Software Bus routing table has become corrupt and is sending messages targeted for other Applications to the cFE Table Services Application.

The `ID` field in the event message identifies the message ID (in hex) that was found in the message.

Definition at line 262 of file cfe_tbl_events.h.

13.65.1.37 CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID

```
#define CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID 66
```

Event Message `'No Inactive Buffer for Table '%s' present'`

Type: ERROR

Cause:

This event message is generated when a Table Dump or a Table Validate command for an Inactive Table Buffer was received and there isn't an Inactive Table Buffer associated with the specified Table.

Definition at line 528 of file cfe_tbl_events.h.

13.65.1.38 CFE_TBL_NO_SUCH_TABLE_ERR_EID

```
#define CFE_TBL_NO_SUCH_TABLE_ERR_EID 57
```

Event Message `'Unable to locate '%s' in Table Registry'`

Type: ERROR

Cause:

This event message is generated when a command that specifies a table name has a table name that is not found in the Table Registry. Most likely causes for this are:

1. Table name was misspelled in the command.
2. The Application that Registered the Table has either failed to run or has been terminated thus removing the Table from the Registry.
3. The Table Registry has become corrupted.

Definition at line 385 of file cfe_tbl_events.h.

13.65.1.39 CFE_TBL_NO_WORK_BUFFERS_ERR_EID

```
#define CFE_TBL_NO_WORK_BUFFERS_ERR_EID 60
```

Event Message `'No working buffers available for table '%s''`

Type: ERROR

Cause:

This event message is generated when either a Table Load Command for a Single Buffered Table or a Table Dump Command for a Dump Only Table has been sent AND there are no Shared Buffers available to hold either the load image or the dump image. To free a Shared Buffer, either a previously loaded table image must be activated or aborted OR the operator has to wait for previously dumped Dump Only tables have had a chance to be written to a file (which occurs whenever the cFE Table Services receives a Housekeeping Request).

Definition at line 443 of file cfe_tbl_events.h.

13.65.1.40 CFE_TBL_NOOP_INF_EID

```
#define CFE_TBL_NOOP_INF_EID 10
```

Event Message `'No-op command'`

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Table Services NO-OP command

Definition at line 83 of file cfe_tbl_events.h.

13.65.1.41 CFE_TBL_NOT_CRITICAL_TBL_ERR_EID

```
#define CFE_TBL_NOT_CRITICAL_TBL_ERR_EID 83
```

Event Message `'Table '%s' is in Critical Table Registry but CDS is not tagged as a table'`

Type: ERROR

Cause:

This event message is generated when a Table Delete Critical Data Store command is received specifying a CDS name for a Critical Data Store that is NOT a critical table image. To delete CDSs that are not Critical Table Images, the Executive Services command CFE_ES_DELETE_CDS_CC must be used.

Definition at line 759 of file cfe_tbl_events.h.

13.65.1.42 CFE_TBL_NOT_IN_CRIT_REG_ERR_EID

```
#define CFE_TBL_NOT_IN_CRIT_REG_ERR_EID 84
```

Event Message `'Table '%s' is not found in Critical Table Registry'`

Type: ERROR

Cause:

This event message is generated when a Table Delete Critical Data Store command is received specifying a table name that cannot be found in the Critical Table Registry. If a Critical Data Store exists with the specified name, then the Critical Table Registry has somehow gotten out of sync with the CDS. Otherwise, the likely cause of this error is a misspelled table name in the command.

Definition at line 773 of file cfe_tbl_events.h.

13.65.1.43 CFE_TBL_OVERWRITE_DUMP_INF_EID

```
#define CFE_TBL_OVERWRITE_DUMP_INF_EID 13
```

Event Message `'Successfully overwrote '%s' with Table '%s''`

Type: INFORMATION

Cause:

This event message is always generated after a successful execution of a cFE Table Services Dump Table command where the command specified target filename was the same as a file already present in the onboard filesystem. If the specified file did not exist, the event message would have been CFE_TBL_WRITE_DUMP_INF_EID.

Definition at line 122 of file cfe_tbl_events.h.

13.65.1.44 CFE_TBL_OVERWRITE_REG_DUMP_INF_EID

```
#define CFE_TBL_OVERWRITE_REG_DUMP_INF_EID 15
```

Event Message `'Successfully overwrote '%s' with Table Registry'`

Type: DEBUG

Cause:

This event message is always generated after a successful execution of a cFE Table Services Dump Table Registry command where the command specified target filename was the same as a file already present in the onboard filesystem. If the specified file did not exist, the event message would have been CFE_TBL_WRITE_REG_DUMP_INF_EID.

Definition at line 152 of file cfe_tbl_events.h.

13.65.1.45 CFE_TBL_PARTIAL_LOAD_ERR_EID

```
#define CFE_TBL_PARTIAL_LOAD_ERR_EID 74
```

Event Message ''%s' has partial load for uninitialized table '%s''

Type: ERROR

Cause:

This event message is generated when a Load Table command is received and the Table Header in the specified Table Image file indicates the starting offset for the table is non-zero and the table has never been previously, completely loaded. Partial Table loads are only allowed after the table has had a successful load.

Definition at line 642 of file cfe_tbl_events.h.

13.65.1.46 CFE_TBL_PROCESSOR_ID_ERR_EID

```
#define CFE_TBL_PROCESSOR_ID_ERR_EID 98
```

Event Message 'Unable to verify Processor ID for '%s', ID = 0x%08X'

Type: ERROR

Cause:

This event message is generated when either an Application calls the CFE_TBL_Load API or a Table Load command has been received and the specified table file has failed Processor ID validation. Verification of Processor ID in table files is enabled/disabled via CFE_PLATFORM_TBL_VALID_PRID_COUNT, defined in the platform configuration header file. This event message can only be generated if CFE_PLATFORM_TBL_VALID_PRID_COUNT has a non-zero value and the table file has a cFE Standard File Header whose Processor ID does not match one of the values defined for Processor ID verification in the platform config file. The most likely causes for this error are:

1. The specified table file is not intended for this processor.
2. The specified table file has been created with bad "endianess" (headers should always conform to a big endian format).
3. The specified table file has become corrupted.
4. The definition for CFE_PLATFORM_TBL_VALID_PRID_COUNT is not large enough to include all of the valid Processor ID entries in the platform config file.
5. There is no entry for this Processor ID in the platform config file list of valid Processor ID's.

The `ID` field specified in the event message contains the Processor ID that was found in the specified table file.

Definition at line 1053 of file cfe_tbl_events.h.

13.65.1.47 CFE_TBL_REGISTER_ERR_EID

```
#define CFE_TBL_REGISTER_ERR_EID 90
```

Event Message `'%s Failed to Register '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Register unsuccessfully.

The `Status` field of the Event Message can be used to identify the reason for the failure by looking it up in the cfe_↩ error.h file

Definition at line 913 of file cfe_tbl_events.h.

13.65.1.48 CFE_TBL_RESET_INF_EID

```
#define CFE_TBL_RESET_INF_EID 11
```

Event Message `'Reset Counters command'`

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Table Services Reset Counters command

Definition at line 95 of file cfe_tbl_events.h.

13.65.1.49 CFE_TBL_SHARE_ERR_EID

```
#define CFE_TBL_SHARE_ERR_EID 91
```

Event Message `'%s Failed to Share '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Share unsuccessfully.

The `Status` field of the Event Message can be used to identify the reason for the failure by looking it up in the cfe_error.h file

Definition at line 927 of file cfe_tbl_events.h.

13.65.1.50 CFE_TBL_SPACECRAFT_ID_ERR_EID

```
#define CFE_TBL_SPACECRAFT_ID_ERR_EID 97
```

Event Message `'Unable to verify Spacecraft ID for '%s', ID = 0x%08X'`

Type: ERROR

Cause:

This event message is generated when either an Application calls the CFE_TBL_Load API or a Table Load command has been received and the specified table file has failed Spacecraft ID validation. Verification of Spacecraft ID in table files is enabled/disabled via CFE_PLATFORM_TBL_VALID_SCID_COUNT, defined in the platform configuration header file. This event message can only be generated if CFE_PLATFORM_TBL_VALID_SCID_COUNT has a non-zero value and the table file has a cFE Standard File Header whose Spacecraft ID does not match one of the values defined for Spacecraft ID verification in the platform config file. The most likely causes for this error are:

1. The specified table file is not intended for this spacecraft.
2. The specified table file has been created with bad "endianess" (headers should always conform to a big endian format).
3. The specified table file has become corrupted.
4. The definition for CFE_PLATFORM_TBL_VALID_SCID_COUNT is not large enough to include all of the valid Spacecraft ID entries in the platform config file.
5. There is no entry for this Spacecraft ID in the platform config file list of valid Spacecraft ID's.

The `ID` field specified in the event message contains the Spacecraft ID that was found in the specified table file.

Definition at line 1025 of file cfe_tbl_events.h.

13.65.1.51 CFE_TBL_TLM_REG_CMD_INF_EID

```
#define CFE_TBL_TLM_REG_CMD_INF_EID 18
```

Event Message `'Table Registry entry for '%s' will be telemetered'`

Type: DEBUG

Cause:

This event message is generated upon successful execution of a cFE Table Services Telemeter Table Registry Entry command . Subsequent Table Services Housekeeping Telemetry should contain the desired Table Registry Entry data.

Definition at line 199 of file cfe_tbl_events.h.

13.65.1.52 CFE_TBL_TOO_MANY_DUMPS_ERR_EID

```
#define CFE_TBL_TOO_MANY_DUMPS_ERR_EID 76
```

Event Message `'Too many Dump Only Table Dumps have been requested'`

Type: ERROR

Cause:

This event message is generated when a Table Dump command for a Dump-Only Table was received and there are no more free Dump Only Control Blocks available. The number of simultaneous Dump Only Tables that can be pending is specified by the configuration parameter CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS which is found in the cfe_platform_cfg.h file.

Definition at line 668 of file cfe_tbl_events.h.

13.65.1.53 CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID

```
#define CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID 67
```

Event Message `'Too many Table Validations have been requested'`

Type: ERROR

Cause:

This event message is generated when a Table Validate command was received and there are no more free Validation Result Blocks available. The number of simultaneous validations that can be pending is specified by the configuration parameter CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS which is found in the cfe_platform_cfg.h file.

Validation Commands lock one of the Validation Result Blocks upon receipt of the validation command until the result of the Validation, performed by the table's owning Application, has been reported in a Table Services Housekeeping Request Message.

Definition at line 546 of file cfe_tbl_events.h.

13.65.1.54 CFE_TBL_UNREGISTER_ERR_EID

```
#define CFE_TBL_UNREGISTER_ERR_EID 92
```

Event Message `'%s Failed to Unregister '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Unregister unsuccessfully.

The `Status` field of the Event Message can be used to identify the reason for the failure by looking it up in the cfe_error.h file

Definition at line 941 of file cfe_tbl_events.h.

13.65.1.55 CFE_TBL_UNVALIDATED_ERR_EID

```
#define CFE_TBL_UNVALIDATED_ERR_EID 81
```

Event Message `'Cannot activate table '%s'.  Inactive image not Validated'`

Type: ERROR

Cause:

This event message is generated when a Table Activate command is received specifying a Table Image that has not been Validated. If a table has a validation function associated with it (as defined by the owning Application when the Table is first Registered), then the Inactive Image MUST be successfully Validated prior to Activation.

Definition at line 733 of file cfe_tbl_events.h.

13.65.1.56 CFE_TBL_UPDATE_ERR_EID

```
#define CFE_TBL_UPDATE_ERR_EID 95
```

Event Message `'%s Failed to Update '%s', Status=0x%08X"`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Update (or, via an internal call, the CFE_TBL_Manage) API and the Table fails to properly update.

The `Status` parameter in the Event Message can be used to identify the reason for the failure by looking it up in the cfe_error.h file.

Definition at line 982 of file cfe_tbl_events.h.

13.65.1.57 CFE_TBL_UPDATE_SUCCESS_INF_EID

```
#define CFE_TBL_UPDATE_SUCCESS_INF_EID 37
```

Event Message `'%s Successfully Updated '%s''`

Type: INFORMATION

Cause:

This event message is generated when a Table's Active Buffer is successfully updated with the contents of its Inactive Buffer.

Definition at line 883 of file cfe_tbl_events.h.

13.65.1.58 CFE_TBL_VAL_REQ_MADE_INF_EID

```
#define CFE_TBL_VAL_REQ_MADE_INF_EID 16
```

Event Message `'Tbl Services issued validation request for '%s''`

Type: DEBUG

Cause:

This event message is generated upon successful execution of a cFE Table Services Validate Table command . It should be noted, however, that this Event Message does *NOT* indicate completion of the Table Validation. It is *ONLY* indicating that the appropriate flag has been set to *NOTIFY* the table's owning Application that a Validation has been requested. Completion of the Validation is indicated by either the CFE_TBL_VALIDATION_INF_EID or CFE_TBL_VALIDATION↩ _ERR_EID event messages.

Definition at line 169 of file cfe_tbl_events.h.

13.65.1.59 CFE_TBL_VALIDATION_ERR_EID

```
#define CFE_TBL_VALIDATION_ERR_EID 96
```

Event Message `'%s validation failed for Inactive '%s', Status=0x%08X"`

Type: ERROR

Cause:

This event message is generated when an Application calls CFE_TBL_Validate (or, via an internal call, the CFE_TBL_Manage) API and the Table fails its Validation.

The `Status` parameter in the Event Message contains the status code returned by the Table's Validation function as defined by the owning Application when the Table was Registered.

Definition at line 997 of file cfe_tbl_events.h.

13.65.1.60 CFE_TBL_VALIDATION_INF_EID

```
#define CFE_TBL_VALIDATION_INF_EID 36
```

Event Message `'%s validation successful for Inactive '%s''`

Type: INFORMATION

Cause:

This event message is generated when a Table Image is successfully validated by its owning Application via the Validation function specified by the owning Application when the table was first registered.

Definition at line 871 of file cfe_tbl_events.h.

13.65.1.61 CFE_TBL_WRITE_CFE_HDR_ERR_EID

```
#define CFE_TBL_WRITE_CFE_HDR_ERR_EID 63
```

Event Message `'Error writing cFE File Header to '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when a Table Dump or Table Registry Dump command was received and the cFE Table Services is unable to write the standard cFE File Header to the specified file.

The `Status` field provides the return status from the CFE_FS_WriteHeader function call.

Definition at line 486 of file cfe_tbl_events.h.

13.65.1.62 CFE_TBL_WRITE_DUMP_INF_EID

```
#define CFE_TBL_WRITE_DUMP_INF_EID 14
```

Event Message `'Successfully dumped Table '%s' to '%s''`

Type: INFORMATION

Cause:

This event message is always generated after a successful execution of a cFE Table Services Dump Table command where the command specified target filename was a currently non-existent file. If the file did already exist, the event message would have been CFE_TBL_OVERWRITE_DUMP_INF_EID.

Definition at line 137 of file cfe_tbl_events.h.

13.65.1.63 CFE_TBL_WRITE_REG_DUMP_INF_EID

```
#define CFE_TBL_WRITE_REG_DUMP_INF_EID 22
```

Event Message `'Successfully dumped Table Registry to '%s':Size=%d,Entries=%d'`

Type: DEBUG

Cause:

This event message is always generated after a successful execution of a cFE Table Services Dump Table Registry command where the command specified target filename was a currently non-existent file. If the file did already exist, the event message would have been CFE_TBL_OVERWRITE_REG_DUMP_INF_EID.

Definition at line 226 of file cfe_tbl_events.h.

13.65.1.64 CFE_TBL_WRITE_TBL_HDR_ERR_EID

```
#define CFE_TBL_WRITE_TBL_HDR_ERR_EID 64
```

Event Message `'Error writing Tbl image File Header to '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when a Table Dump command was received and the cFE Table Services is unable to write the standard cFE Table Image Header to the specified file.

The `Status` field provides the return status from the OS_write function call.

Definition at line 500 of file cfe_tbl_events.h.

13.65.1.65 CFE_TBL_WRITE_TBL_IMG_ERR_EID

```
#define CFE_TBL_WRITE_TBL_IMG_ERR_EID 65
```

Event Message `'Error writing Tbl image to '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when a Table Dump command was received and the cFE Table Services is unable to write the contents of the specified Table image to the specified file.

The `Status` field provides the return status from the OS_write function call.

Definition at line 515 of file cfe_tbl_events.h.

13.65.1.66 CFE_TBL_WRITE_TBL_REG_ERR_EID

```
#define CFE_TBL_WRITE_TBL_REG_ERR_EID 68
```

Event Message `'Error writing Registry to '%s', Status=0x%08X'`

Type: ERROR

Cause:

This event message is generated when a Table Registry Dump command was received and the cFE Table Services is unable to write the entire contents of the Table Registry to the specified file.

The `Status` field provides the return status from the OS_write function call.

Definition at line 561 of file cfe_tbl_events.h.

13.65.1.67 CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID

```
#define CFE_TBL_ZERO_LENGTH_LOAD_ERR_EID 73
```

Event Message `'Table Hdr in '%s' indicates no data in file'`

Type: ERROR

Cause:

This event message is generated when a Load Table command is received and the Table Header in the specified Table Image file claims the file contains no data.

Definition at line 628 of file cfe_tbl_events.h.

13.66 cfe/fsw/cfe-core/src/inc/cfe_tbl_extern_typedefs.h File Reference

```
#include "common_types.h"
#include <cfe_mission_cfg.h>
```

Data Structures

- struct CFE_TBL_File_Hdr_t

  The definition of the header fields that are included in CFE Table Data files.

Typedefs

- typedef uint16 CFE_TBL_BufferSelect_Enum_t

  Selects the buffer to operate on for validate or dump commands.

Enumerations

- enum CFE_TBL_BufferSelect { CFE_TBL_BufferSelect_INACTIVE = 0, CFE_TBL_BufferSelect_ACTIVE = 1 }

  Label definitions associated with CFE_TBL_BufferSelect_Enum_t.

13.66.1 Typedef Documentation

13.66.1.1 CFE_TBL_BufferSelect_Enum_t

```
typedef uint16 CFE_TBL_BufferSelect_Enum_t
```

See also

enum CFE_TBL_BufferSelect

Definition at line 60 of file cfe_tbl_extern_typedefs.h.

13.66.2 Enumeration Type Documentation

13.66.2.1 CFE_TBL_BufferSelect

```
enum CFE_TBL_BufferSelect
```

Enumerator

CFE_TBL_BufferSelect_INACTIVE	Select the Inactive buffer for validate or dump.
CFE_TBL_BufferSelect_ACTIVE	Select the Active buffer for validate or dump.

Definition at line 40 of file cfe_tbl_extern_typedefs.h.

13.67 cfe/fsw/cfe-core/src/inc/cfe_tbl_filedef.h File Reference

```
#include <cfe_mission_cfg.h>
#include <common_types.h>
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_fs_extern_typedefs.h"
```

Data Structures

- struct CFE_TBL_FileDef_t

Macros

- #define CFE_TBL_FILEDEF(ObjName, TblName, Desc, Filename) static OS_USED CFE_TBL_FileDef_t CFE↩ _TBL_FileDef={#ObjName, #TblName, #Desc, #Filename, sizeof(ObjName)};

13.67.1 Macro Definition Documentation

13.67.1.1 CFE_TBL_FILEDEF

```
#define CFE_TBL_FILEDEF(
                ObjName,
                TblName,
                Desc,
                Filename ) static OS_USED CFE_TBL_FileDef_t CFE_TBL_FileDef={#ObjName, #TblName,
#Desc, #Filename, sizeof(ObjName)};
```

The CFE_TBL_FILEDEF macro can be used to simplify the declaration of a table image when using the elf2cfetbl utility. An example of the source code and how this macro would be used is as follows:

```
 #include "cfe_tbl_filedef.h"

typedef struct
{
    int     Int1;
    int     Int2;
    int     Int3;
    char    Char1;
} MyTblStruct_t;

MyTblStruct_t MyTblStruct = { 0x01020304, 0x05060708, 0x090A0B0C, 0x0D };

CFE_TBL_FILEDEF(MyTblStruct, MyApp.TableName, Table Utility Test Table, MyTblDefault.bin )
```

Definition at line 90 of file cfe_tbl_filedef.h.

13.68 cfe/fsw/cfe-core/src/inc/cfe_tbl_msg.h File Reference

```
#include "cfe.h"
```

Data Structures

- struct CFE_TBL_NoArgsCmd_t

  Generic "no arguments" command.
- struct CFE_TBL_LoadCmd_Payload_t

  Load Table Command.
- struct CFE_TBL_Load_t
- struct CFE_TBL_DumpCmd_Payload_t

  Dump Table Command.
- struct CFE_TBL_Dump_t
- struct CFE_TBL_ValidateCmd_Payload_t

  Validate Table Command.
- struct CFE_TBL_Validate_t
- struct CFE_TBL_ActivateCmd_Payload_t

Activate Table Command.

- struct CFE_TBL_Activate_t
- struct CFE_TBL_DumpRegistryCmd_Payload_t

Dump Registry Command.

- struct CFE_TBL_DumpRegistry_t
- struct CFE_TBL_SendRegistryCmd_Payload_t

Telemeter Table Registry Entry Command.

- struct CFE_TBL_SendRegistry_t
- struct CFE_TBL_DelCDSCmd_Payload_t

Delete Critical Table CDS Command.

- struct CFE_TBL_DeleteCDS_t
- struct CFE_TBL_AbortLoadCmd_Payload_t

Abort Load Command.

- struct CFE_TBL_AbortLoad_t
- struct CFE_TBL_NotifyCmd_Payload_t

Table Management Notification Message.

- struct CFE_TBL_NotifyCmd_t
- struct CFE_TBL_HousekeepingTlm_Payload_t
- struct CFE_TBL_HousekeepingTlm_t
- struct CFE_TBL_TblRegPacket_Payload_t
- struct CFE_TBL_TableRegistryTlm_t

Macros

Table Services Command Codes

- #define CFE_TBL_NOOP_CC 0
- #define CFE_TBL_RESET_COUNTERS_CC 1
- #define CFE_TBL_LOAD_CC 2
- #define CFE_TBL_DUMP_CC 3
- #define CFE_TBL_VALIDATE_CC 4
- #define CFE_TBL_ACTIVATE_CC 5
- #define CFE_TBL_DUMP_REGISTRY_CC 6
- #define CFE_TBL_SEND_REGISTRY_CC 7
- #define CFE_TBL_DELETE_CDS_CC 8
- #define CFE_TBL_ABORT_LOAD_CC 9

Typedefs

- typedef CFE_TBL_NoArgsCmd_t CFE_TBL_Noop_t
- typedef CFE_TBL_NoArgsCmd_t CFE_TBL_ResetCounters_t
- typedef CFE_TBL_HousekeepingTlm_t CFE_TBL_HkPacket_t
- typedef CFE_TBL_TableRegistryTlm_t CFE_TBL_TblRegPacket_t

13.68.1 Macro Definition Documentation

13.68.1.1 CFE_TBL_ABORT_LOAD_CC

```
#define CFE_TBL_ABORT_LOAD_CC 9
```

Name Abort Table Load

Description

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. For single buffered tables, the allocated shared working buffer is freed and becomes available for other Table Load commands.

Command Mnemonic(s) **`$sc_$cpu_TBL_LOADABORT`**

Command Structure

CFE_TBL_AbortLoad_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- The CFE_TBL_LOAD_ABORT_INF_EID informational event message is generated
- If the load was aborted for a single buffered table, the **`$sc_$cpu_TBL_NumFreeShrBuf`** **telemetry** point should increment

Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the table registry.
- The specified table did not have a load in progress to be aborted.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TBL_CMDEC`** - command error counter will increment
- Error specific event message

Criticality

This command will cause the loss of data put into an inactive table buffer.

See also

CFE_TBL_LOAD_CC, CFE_TBL_DUMP_CC, CFE_TBL_VALIDATE_CC, CFE_TBL_ACTIVATE_CC

Definition at line 476 of file cfe_tbl_msg.h.

13.68.1.2 CFE_TBL_ACTIVATE_CC

```
#define CFE_TBL_ACTIVATE_CC 5
```

Name Activate Table

Description

This command will cause Table Services to notify a table's owner that an update is pending. The owning application will then update the contents of the active table buffer with the contents of the associated inactive table buffer at a time of their convenience.

Command Mnemonic(s) **`$sc_$cpu_TBL_ACTIVATE`**

Command Structure

CFE_TBL_Activate_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- The CFE_TBL_UPDATE_SUCCESS_INF_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- A single buffered table's inactive buffer was requested to be dumped and no such buffer is currently allocated.
- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TBL_CMDEC`** - command error counter will increment
- Command specific error event message are issued for all error cases

Criticality

This command will cause the contents of the specified table to be updated with the contents in the inactive table buffer.

See also

CFE_TBL_LOAD_CC, CFE_TBL_DUMP_CC, CFE_TBL_VALIDATE_CC, CFE_TBL_ABORT_LOAD_CC

Definition at line 316 of file cfe_tbl_msg.h.

13.68.1.3 CFE_TBL_DELETE_CDS_CC

```
#define CFE_TBL_DELETE_CDS_CC 8
```

Name Delete Critical Table from Critical Data Store

Description

This command will delete the Critical Data Store (CDS) associated with the specified Critical Table. Note that any table still present in the Table Registry is unable to be deleted from the Critical Data Store. All Applications that are accessing the critical table must release and unregister their access before the CDS can be deleted.

Command Mnemonic(s) **`$sc_$cpu_TBL_DeleteCDS`**

Command Structure

CFE_TBL_DeleteCDS_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- The CFE_TBL_CDS_DELETED_INFO_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the critical data store registry
- The specified table name WAS found in the table registry (all registrations/sharing of the table must be unregistered before the table's CDS can be deleted)
- The table's owning application is still active

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TBL_CMDEC`** - command error counter will increment
- Error specific event message

Criticality

This command will cause the loss of the specified table's contents before the owning Application was terminated.

See also

CFE_ES_DUMP_CDS_REGISTRY_CC, CFE_ES_DELETE_CDS_CC

Definition at line 437 of file cfe_tbl_msg.h.

13.68.1.4 CFE_TBL_DUMP_CC

```
#define CFE_TBL_DUMP_CC 3
```

Name Dump Table

Description

This command will cause the Table Services to put the contents of the specified table buffer into the command specified file.

Command Mnemonic(s) **$sc_$cpu_TBL_DUMP**

Command Structure

CFE_TBL_Dump_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TBL_CMDPC -** command execution counter will increment
- Either the CFE_TBL_OVERWRITE_DUMP_INF_EID OR the CFE_TBL_WRITE_DUMP_INF_EID informational event message will be generated

Error Conditions

This command may fail for the following reason(s):

- A single buffered table's inactive buffer was requested to be dumped and no such buffer is currently allocated.
- Error occurred during write operation to file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.
- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_TBL_CMDEC -** command error counter will increment
- A command specific error event message is issued for all error cases

Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_TBL_LOAD_CC, CFE_TBL_VALIDATE_CC, CFE_TBL_ACTIVATE_CC, CFE_TBL_ABORT_LOAD_CC

Definition at line 219 of file cfe_tbl_msg.h.

13.68.1.5 CFE_TBL_DUMP_REGISTRY_CC

```
#define CFE_TBL_DUMP_REGISTRY_CC 6
```

Name Dump Table Registry

Description

This command will cause Table Services to write some of the contents of the Table Registry to the command specified file. This allows the operator to see the current state and configuration of all tables that have been registered with the cFE.

Command Mnemonic(s) **`$sc_$cpu_TBL_WriteReg2File`**

Command Structure

CFE_TBL_DumpRegistry_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- The generation of either CFE_TBL_OVERWRITE_REG_DUMP_INF_EID or CFE_TBL_WRITE_REG_DUMP_INF_EID debug event messages
- The specified file should appear (or be updated) at the specified location in the file system

Error Conditions

This command may fail for the following reason(s):

- Error occurred during write operation to file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TBL_CMDEC`** - command error counter will increment
- An Error specific event message

Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

See also

CFE_TBL_SEND_REGISTRY_CC

Definition at line 358 of file cfe_tbl_msg.h.

13.68.1.6 CFE_TBL_LOAD_CC

```
#define CFE_TBL_LOAD_CC 2
```

Name Load Table

Description

This command loads the contents of the specified file into an inactive buffer for the table specified within the file.

Command Mnemonic(s) **$sc_$cpu_TBL_Load**

Command Structure

CFE_TBL_Load_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TBL_CMDPC -** command execution counter will increment
- The CFE_TBL_FILE_LOADED_INF_EID informational event message will be generated

Error Conditions

This command can fail for the following reasons:

- Table name found in table image file's table header is not found in table registry (ie - The table associated with the table image in the file has not been registered by an application).
- The table image file's header indicates the file contains 'x' number of bytes of data but the file contains less.
- No working buffers are available for the load. This would indicate that too many single-buffered table loads are in progress at the same time.
- The table image file's header indicates the data to be loaded is beyond the size of the table. Either the number of bytes in the file are too many or the starting offset into the table is too high.
- The table image file's header indicates there is no data in the file (ie - Number of bytes to load is zero).
- An attempt is being made to load an uninitialized table with a file containing only a partial table image.
- The table image file was unable to be opened. Either the file does not exist at the specified location, the filename is in error, or the file system has been corrupted.

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_TBL_CMDEC -** command error counter will increment
- Command specific error event messages are issued for all error cases

Criticality

This command is not inherently dangerous. It is performing the first step of loading a table and can be aborted (using the Abort Table Load command described below) without affecting the contents of the active table image.

See also

CFE_TBL_DUMP_CC, CFE_TBL_VALIDATE_CC, CFE_TBL_ACTIVATE_CC, CFE_TBL_ABORT_LOAD_CC

Definition at line 176 of file cfe_tbl_msg.h.

13.68.1.7 CFE_TBL_NOOP_CC

```
#define CFE_TBL_NOOP_CC 0
```

Name Table No-Op

Description

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Table Services task.

Command Mnemonic(s) **`$sc_$cpu_TBL_NOOP`**

Command Structure

CFE_TBL_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- The CFE_TBL_NOOP_INF_EID informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Table Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

None

See also

Definition at line 82 of file cfe_tbl_msg.h.

13.68.1.8 CFE_TBL_RESET_COUNTERS_CC

```
#define CFE_TBL_RESET_COUNTERS_CC 1
```

Name Table Reset Counters

Description

This command resets the following counters within the Table Services housekeeping telemetry:

- Command Execution Counter ($sc_$cpu_TBL_CMDPC)
- Command Error Counter ($sc_$cpu_TBL_CMDEC)
- Successful Table Validations Counter ($sc_$cpu_TBL_ValSuccessCtr)
- Failed Table Validations Counter ($sc_$cpu_TBL_ValFailedCtr)
- Number of Table Validations Requested ($sc_$cpu_TBL_ValReqCtr)

Command Mnemonic(s) **`$sc_$cpu_TBL_ResetCtrs`**

Command Structure

CFE_TBL_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- The CFE_TBL_RESET_INF_EID debug event message will be generated

Error Conditions

There are no error conditions for this command. If the Table Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

See also

Definition at line 122 of file cfe_tbl_msg.h.

13.68.1.9 CFE_TBL_SEND_REGISTRY_CC

```
#define CFE_TBL_SEND_REGISTRY_CC 7
```

Name Telemeter One Table Registry Entry

Description

This command will cause Table Services to telemeter the contents of the Table Registry for the command specified table.

Command Mnemonic(s) **`$sc_$cpu_TBL_TLMReg`**

Command Structure

CFE_TBL_DumpRegistry_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TBL_CMDPC`** - command execution counter will increment
- Receipt of a Table Registry Info Packet (see CFE_TBL_TableRegistryTlm_t)
- The CFE_TBL_TLM_REG_CMD_INF_EID debug event message will be generated

Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TBL_CMDEC`** - command error counter will increment
- Error specific event message

Criticality

This command is not inherently dangerous. It will generate additional telemetry.

See also

CFE_TBL_DUMP_REGISTRY_CC

Definition at line 393 of file cfe_tbl_msg.h.

13.68.1.10 CFE_TBL_VALIDATE_CC

```
#define CFE_TBL_VALIDATE_CC 4
```

Name Validate Table

Description

This command will cause Table Services to calculate the Data Integrity Value for the specified table and to notify the owning application that the table's validation function should be executed. The results of both the Data Integrity Value computation and the validation function are reported in Table Services Housekeeping Telemetry.

Command Mnemonic(s) **$sc_$cpu_TBL_VALIDATE**

Command Structure

CFE_TBL_Validate_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TBL_CMDPC -** command execution counter will increment
- **$sc_$cpu_TBL_ValReqCtr -** table validation request counter will increment
- **$sc_$cpu_TBL_LastValCRC -** calculated data integrity value will be updated
- The CFE_TBL_VAL_REQ_MADE_INF_EID debug event message (indicating the application is being notified of a validation request)

If the specified table has an associated validation function, then the following telemetry will also change:

- Either **$sc_$cpu_TBL_ValSuccessCtr OR $sc_$cpu_TBL_ValFailedCtr will** increment
- **$sc_$cpu_TBL_ValCompltdCtr -** table validations performed counter will increment
- **$sc_$cpu_TBl_LastValS -** table validation function return status will update
- The CFE_TBL_VALIDATION_INF_EID informational event message (indicating the validation function return status) will be generated

Error Conditions

This command may fail for the following reason(s):

- A single buffered table's inactive buffer was requested to be dumped and no such buffer is currently allocated.
- Too many validations have been requested simultaneously. The operator must wait for one or more applications to perform their table validation functions before trying again.
- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_TBL_CMDEC -** command error counter will increment
- Command specific error event message are issued for all error cases

Criticality

The success or failure of a table validation does not have any immediate impact on table contents. The results are sent to the operator in telemetry and the operator must determine whether the results are acceptable and send a command to activate the validated table image.

See also

CFE_TBL_LOAD_CC, CFE_TBL_DUMP_CC, CFE_TBL_ACTIVATE_CC, CFE_TBL_ABORT_LOAD_CC

Definition at line 276 of file cfe_tbl_msg.h.

13.68.2 Typedef Documentation

13.68.2.1 CFE_TBL_HkPacket_t

```
typedef CFE_TBL_HousekeepingTlm_t CFE_TBL_HkPacket_t
```

Definition at line 828 of file cfe_tbl_msg.h.

13.68.2.2 CFE_TBL_Noop_t

```
typedef CFE_TBL_NoArgsCmd_t CFE_TBL_Noop_t
```

Definition at line 504 of file cfe_tbl_msg.h.

13.68.2.3 CFE_TBL_ResetCounters_t

```
typedef CFE_TBL_NoArgsCmd_t CFE_TBL_ResetCounters_t
```

Definition at line 505 of file cfe_tbl_msg.h.

13.68.2.4 CFE_TBL_TblRegPacket_t

```
typedef CFE_TBL_TableRegistryTlm_t CFE_TBL_TblRegPacket_t
```

Definition at line 829 of file cfe_tbl_msg.h.

13.69 cfe/fsw/cfe-core/src/inc/cfe_time.h File Reference

```
#include "cfe_time_extern_typedefs.h"
#include "common_types.h"
```

Data Structures

- struct CFE_TIME_SysTime_t

  Data structure used to hold system time values.
- struct CFE_TIME_ResetVars_t

  Time related variables that are maintained through a Processor Reset.

Macros

- #define CFE_TIME_PRINTED_STRING_SIZE 24

  Required size of buffer to be passed into CFE_TIME_Print (includes null terminator)
- #define CFE_TIME_USE_INTERN CFE_TIME_SourceSelect_INTERNAL
- #define CFE_TIME_USE_EXTERN CFE_TIME_SourceSelect_EXTERNAL
- #define CFE_TIME_TONE_PRI CFE_TIME_ToneSignalSelect_PRIMARY
- #define CFE_TIME_TONE_RED CFE_TIME_ToneSignalSelect_REDUNDANT
- #define CFE_TIME_ADD_ADJUST CFE_TIME_AdjustDirection_ADD
- #define CFE_TIME_SUB_ADJUST CFE_TIME_AdjustDirection_SUBTRACT
- #define CFE_TIME_NO_FLY CFE_TIME_FlywheelState_NO_FLY
- #define CFE_TIME_IS_FLY CFE_TIME_FlywheelState_IS_FLY
- #define CFE_TIME_NOT_SET CFE_TIME_SetState_NOT_SET
- #define CFE_TIME_WAS_SET CFE_TIME_SetState_WAS_SET
- #define CFE_TIME_INVALID CFE_TIME_ClockState_INVALID
- #define CFE_TIME_VALID CFE_TIME_ClockState_VALID
- #define CFE_TIME_FLYWHEEL CFE_TIME_ClockState_FLYWHEEL
- #define CFE_TIME_Copy(m, t) { (m)->Seconds = (t)->Seconds; (m)->Subseconds = (t)->Subseconds; }

Typedefs

- typedef int32(* CFE_TIME_SynchCallbackPtr_t) (void)

  Time Synchronization Callback Function Ptr Type.

Enumerations

- enum CFE_TIME_Compare_t { CFE_TIME_A_LT_B = -1, CFE_TIME_EQUAL = 0, CFE_TIME_A_GT_B = 1 }

  Enumerated types identifying the relative relationships of two times.

Functions

- CFE_TIME_SysTime_t CFE_TIME_GetTime (void)

  Get the current spacecraft time.
- CFE_TIME_SysTime_t CFE_TIME_GetTAI (void)

  Get the current TAI time.
- CFE_TIME_SysTime_t CFE_TIME_GetUTC (void)

  Get the current UTC time.
- CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (CFE_TIME_SysTime_t METTime)

  Convert specified MET into Spacecraft Time.
- CFE_TIME_SysTime_t CFE_TIME_GetSTCF (void)

  Get the current value of the spacecraft time correction factor (STCF).
- CFE_TIME_SysTime_t CFE_TIME_GetMET (void)

  Get the current value of the Mission Elapsed Time (MET).
- uint32 CFE_TIME_GetMETseconds (void)

  Get the current seconds count of the mission-elapsed time.
- uint32 CFE_TIME_GetMETsubsecs (void)

  Get the current sub-seconds count of the mission-elapsed time.
- int16 CFE_TIME_GetLeapSeconds (void)

  Get the current value of the leap seconds counter.
- CFE_TIME_ClockState_Enum_t CFE_TIME_GetClockState (void)

  Get the current state of the spacecraft clock.
- uint16 CFE_TIME_GetClockInfo (void)

  Provides information about the spacecraft clock.
- CFE_TIME_SysTime_t CFE_TIME_Add (CFE_TIME_SysTime_t Time1, CFE_TIME_SysTime_t Time2)

  Adds two time values.
- CFE_TIME_SysTime_t CFE_TIME_Subtract (CFE_TIME_SysTime_t Time1, CFE_TIME_SysTime_t Time2)

  Subtracts two time values.
- CFE_TIME_Compare_t CFE_TIME_Compare (CFE_TIME_SysTime_t TimeA, CFE_TIME_SysTime_t TimeB)

  Compares two time values.
- uint32 CFE_TIME_Sub2MicroSecs (uint32 SubSeconds)

  Converts a sub-seconds count to an equivalent number of microseconds.
- uint32 CFE_TIME_Micro2SubSecs (uint32 MicroSeconds)

  Converts a number of microseconds to an equivalent sub-seconds count.
- uint32 CFE_TIME_CFE2FSSeconds (uint32 SecondsCFE)

  Converts cFE seconds into the File System's seconds.
- uint32 CFE_TIME_FS2CFESeconds (uint32 SecondsFS)

  Converts a file system's seconds into cFE seconds.
- void CFE_TIME_Print (char *PrintBuffer, CFE_TIME_SysTime_t TimeToPrint)

  Print a time value as a string.
- void CFE_TIME_ExternalTone (void)

  Provides the 1 Hz signal from an external source.
- void CFE_TIME_ExternalMET (CFE_TIME_SysTime_t NewMET)

  Provides the Mission Elapsed Time from an external source.
- void CFE_TIME_ExternalGPS (CFE_TIME_SysTime_t NewTime, int16 NewLeaps)

  Provide the time from an external source that has data common to GPS receivers.
- void CFE_TIME_ExternalTime (CFE_TIME_SysTime_t NewTime)

Provide the time from an external source that measures time relative to a known epoch.

- int32 CFE_TIME_RegisterSynchCallback (CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr)

  Registers a callback function that is called whenever time synchronization occurs.
- int32 CFE_TIME_UnregisterSynchCallback (CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr)

  Unregisters a callback function that is called whenever time synchronization occurs.
- void CFE_TIME_Local1HzISR (void)

  This function should be called from the system PSP layer once per second.

13.69.1 Macro Definition Documentation

13.69.1.1 CFE_TIME_ADD_ADJUST

```
#define CFE_TIME_ADD_ADJUST CFE_TIME_AdjustDirection_ADD
```

Definition at line 74 of file cfe_time.h.

13.69.1.2 CFE_TIME_Copy

```
#define CFE_TIME_Copy(
              m,
              t ) { (m)->Seconds = (t)->Seconds; (m)->Subseconds = (t)->Subseconds; }
```

Definition at line 125 of file cfe_time.h.

13.69.1.3 CFE_TIME_FLYWHEEL

```
#define CFE_TIME_FLYWHEEL CFE_TIME_ClockState_FLYWHEEL
```

Definition at line 94 of file cfe_time.h.

13.69.1.4 CFE_TIME_INVALID

```
#define CFE_TIME_INVALID CFE_TIME_ClockState_INVALID
```

Definition at line 92 of file cfe_time.h.

13.69.1.5 CFE_TIME_IS_FLY

```
#define CFE_TIME_IS_FLY CFE_TIME_FlywheelState_IS_FLY
```

Definition at line 81 of file cfe_time.h.

13.69.1.6 CFE_TIME_NO_FLY

```
#define CFE_TIME_NO_FLY CFE_TIME_FlywheelState_NO_FLY
```

Definition at line 80 of file cfe_time.h.

13.69.1.7 CFE_TIME_NOT_SET

```
#define CFE_TIME_NOT_SET CFE_TIME_SetState_NOT_SET
```

Definition at line 86 of file cfe_time.h.

13.69.1.8 CFE_TIME_PRINTED_STRING_SIZE

```
#define CFE_TIME_PRINTED_STRING_SIZE 24
```

Definition at line 49 of file cfe_time.h.

Referenced by CFE_ES_SysLog_vsnprintf(), and CFE_ES_SysLogAppend_Unsync().

13.69.1.9 CFE_TIME_SUB_ADJUST

```
#define CFE_TIME_SUB_ADJUST CFE_TIME_AdjustDirection_SUBTRACT
```

Definition at line 75 of file cfe_time.h.

13.69.1.10 CFE_TIME_TONE_PRI

```
#define CFE_TIME_TONE_PRI CFE_TIME_ToneSignalSelect_PRIMARY
```

Definition at line 68 of file cfe_time.h.

13.69.1.11 CFE_TIME_TONE_RED

```
#define CFE_TIME_TONE_RED CFE_TIME_ToneSignalSelect_REDUNDANT
```

Definition at line 69 of file cfe_time.h.

13.69.1.12 CFE_TIME_USE_EXTERN

```
#define CFE_TIME_USE_EXTERN CFE_TIME_SourceSelect_EXTERNAL
```

Definition at line 63 of file cfe_time.h.

13.69.1.13 CFE_TIME_USE_INTERN

```
#define CFE_TIME_USE_INTERN CFE_TIME_SourceSelect_INTERNAL
```

Definition at line 62 of file cfe_time.h.

13.69.1.14 CFE_TIME_VALID

```
#define CFE_TIME_VALID CFE_TIME_ClockState_VALID
```

Definition at line 93 of file cfe_time.h.

13.69.1.15 CFE_TIME_WAS_SET

```
#define CFE_TIME_WAS_SET CFE_TIME_SetState_WAS_SET
```

Definition at line 87 of file cfe_time.h.

13.69.2 Typedef Documentation

13.69.2.1 CFE_TIME_SynchCallbackPtr_t

```
typedef int32(* CFE_TIME_SynchCallbackPtr_t) (void)
```

Description

Applications that wish to get direct notification of the receipt of the cFE Time Synchronization signal (typically a 1 Hz signal), must register a callback function with the following prototype via the CFE_TIME_RegisterSynchCallback API.

Definition at line 175 of file cfe_time.h.

13.69.3 Enumeration Type Documentation

13.69.3.1 CFE_TIME_Compare_t

```
enum CFE_TIME_Compare_t
```

Description

Since time fields contain numbers that are relative to an epoch time, then it is possible for a time value to be "negative". This can lead to some confusion about what relationship exists between two time values. To resolve this confusion, the cFE provides the API CFE_TIME_Compare which returns these enumerated values.

Enumerator

CFE_TIME_A_LT_B	The first specified time is considered to be before the second specified time.
CFE_TIME_EQUAL	The two specified times are considered to be equal.
CFE_TIME_A_GT_B	The first specified time is considered to be after the second specified time.

Definition at line 138 of file cfe_time.h.

13.69.4 Function Documentation

13.69.4.1 CFE_TIME_Add()

```
CFE_TIME_SysTime_t CFE_TIME_Add (
            CFE_TIME_SysTime_t Time1,
            CFE_TIME_SysTime_t Time2 )
```

Description

This routine adds the two specified times and returns the result. Normally, at least one of the input times should be a value representing a delta time. Adding two absolute times together will not cause an error, but the result will probably be meaningless.

Assumptions, External Events, and Notes:

None

Parameters

in	*Time1*	The first time to be added.
in	*Time2*	The second time to be added.

The sum of the two times, in the CFE_TIME_SysTime_t format described above. If the sum is greater than the maximum value that can be stored in a CFE_TIME_SysTime_t, the result will roll over (this is not considered an error).

Returns

See also

CFE_TIME_Subtract, CFE_TIME_Compare

13.69.4.2 CFE_TIME_CFE2FSSeconds()

```
uint32 CFE_TIME_CFE2FSSeconds (
            uint32 SecondsCFE )
```

Description

File systems use specific time epochs for their time tagging of files. Since spacecraft systems rarely use an epoch that matches a particular file system, this function provides a mechanism to translate a given spacecraft time (in seconds) to the file system's time. The conversion is controlled by the configuration parameter CFE_MISSIO↩N_TIME_FS_FACTOR which is set equal to the number of seconds between the spacecraft's epoch and the file system's epoch.

Assumptions, External Events, and Notes:

None

Parameters

in	*SecondsCFE*	The spacecraft time, in seconds, to be converted.

The equivalent time, in seconds, for the file system.

Returns

See also

CFE_TIME_MET2SCTime, CFE_TIME_Sub2MicroSecs, CFE_TIME_Micro2SubSecs, CFE_TIME_FS2CFE↩ Seconds

13.69.4.3 CFE_TIME_Compare()

```
CFE_TIME_Compare_t CFE_TIME_Compare (
            CFE_TIME_SysTime_t TimeA,
            CFE_TIME_SysTime_t TimeB )
```

Description

This routine compares two time values to see which is "greater". It is important that applications use this function rather than trying to directly compare the component pieces of times. This function will handle roll-over cases seamlessly, which may not be intuitively obvious. The cFE's internal representation of time "rolls over" when the 32 bit seconds count reaches 0xFFFFFFFF. Also, subtracting a delta time from an absolute time close to the epoch could result in "roll under". The strange cases that result from these situations can be handled by defining the comparison function for times as follows: Plot the two times on the circumference of a circle where 0 is at the top and 0x80000000 is at the bottom. If the shortest arc from time A to time B runs clockwise around the circle, then time A is less than time B. If the shortest arc from A to B runs counter-clockwise, then time A is greater than time B.

Assumptions, External Events, and Notes:

None

Parameters

in	*TimeA*	The first time to compare.
in	*TimeB*	The second time to compare.

The result of comparing the two times, one of:	
CFE_TIME_EQUAL	The two specified times are considered to be equal.
CFE_TIME_A_GT↩ _B	The first specified time is considered to be after the second specified time.
CFE_TIME_A_LT↩ _B	The first specified time is considered to be before the second specified time.

Returns

See also

CFE_TIME_Add, CFE_TIME_Subtract

13.69.4.4 CFE_TIME_ExternalGPS()

```
void CFE_TIME_ExternalGPS (
            CFE_TIME_SysTime_t NewTime,
            int16 NewLeaps )
```

Description

This routine provides a method to provide cFE TIME with current time data acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration parameter specified window for tone signal and data packet verification.

Internally, cFE TIME will calculate a new STCF as the difference between this new time value and the spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is CFE_PLATFORM_TIME_CFG_SERVER which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is CFE_PLATFORM_TIME_CFG_SOURCE which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is CFE_PLATFORM_TIME_CFG_SRC_GPS, which indicates that the external time data consists of a time value relative to a known epoch, plus a leap seconds value.

Parameters

in	*NewTime*	The MET value at the next (or previous) 1 Hz tone signal.
in	*NewLeaps*	The Leap Seconds value used to calculate time as UTC.

See also

CFE_TIME_ExternalTone, CFE_TIME_ExternalMET, CFE_TIME_ExternalTime

13.69.4.5 CFE_TIME_ExternalMET()

```
void CFE_TIME_ExternalMET (
            CFE_TIME_SysTime_t NewMET )
```

Description

This routine provides a method to provide cFE TIME with MET acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone"

data command must arrive within the configuration parameter specified window for tone signal and data packet verification.

The MET value at the tone "should" have zero subseconds. Although the interface accepts non-zero values for sub-seconds, it may be harmful to other applications that expect zero subseconds at the moment of the tone. Any decision to use non-zero subseconds should be carefully considered.

Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is CFE_PLATFORM_TIME_CFG_SERVER which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is CFE_PLATFORM_TIME_CFG_SOURCE which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is CFE_PLATFORM_TIME_CFG_SRC_MET, which indicates that the external time data consists of MET.

Parameters

in	*NewMET*	The MET value at the next (or previous) 1 Hz tone signal.

See also

CFE_TIME_ExternalTone, CFE_TIME_ExternalGPS, CFE_TIME_ExternalTime

13.69.4.6 CFE_TIME_ExternalTime()

```
void CFE_TIME_ExternalTime (
            CFE_TIME_SysTime_t NewTime )
```

Description

This routine provides a method to provide cFE TIME with current time data acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration specified window for tone signal and data packet verification.

Internally, cFE TIME will calculate a new STCF as the difference between this new time value and the spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is CFE_PLATFORM_TIME_CFG_SERVER which defines this instanciation of cFE TIME as a time server (not a client). The second required configuration parameter is CFE_PLATFORM_TIME_CFG_SOURCE which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is CFE_PLATFORM_TIME_CFG_SRC_TIME, which indicates that the external time data consists of a time value relative to a known epoch.

Parameters

in	*NewTime*	The MET value at the next (or previous) 1 Hz tone signal.

See also

CFE_TIME_ExternalTone, CFE_TIME_ExternalMET, CFE_TIME_ExternalGPS

13.69.4.7 CFE_TIME_ExternalTone()

```
void CFE_TIME_ExternalTone (
            void  )
```

Description

This routine provides a method for cFE TIME software to be notified of the occurance of the 1Hz tone signal without knowledge of the specific hardware design. Regardless of the source of the tone, this routine should be called as soon as possible after detection to allow cFE TIME software the opportunity to latch the local clock as close as possible to the instant of the tone.

Assumptions, External Events, and Notes:

- This routine may be called directly from within the context of an interrupt handler.

See also

CFE_TIME_ExternalMET, CFE_TIME_ExternalGPS, CFE_TIME_ExternalTime

13.69.4.8 CFE_TIME_FS2CFESeconds()

```
uint32 CFE_TIME_FS2CFESeconds (
            uint32 SecondsFS )
```

Description

File systems use specific time epochs for their time tagging of files. Since spacecraft systems rarely use an epoch that matches a particular file system, this function provides a mechanism to translate a file system time (in seconds) into the spacecraft time (in seconds). The conversion is controlled by the configuration parameter CFE_MISSION_TIME_FS_FACTOR which is set equal to the number of seconds between the spacecraft's epoch and the file system's epoch.

Assumptions, External Events, and Notes:

None

Parameters

in	*SecondsFS*	The file system time, in seconds, to be converted.

The equivalent time, in seconds, for the spacecraft.

Returns

See also

CFE_TIME_MET2SCTime, CFE_TIME_Sub2MicroSecs, CFE_TIME_Micro2SubSecs, CFE_TIME_CFE2FS↩ Seconds

13.69.4.9 CFE_TIME_GetClockInfo()

```
uint16 CFE_TIME_GetClockInfo (
            void  )
```

Description

This routine returns information on the spacecraft clock in a bit mask.

Assumptions, External Events, and Notes:

None

Spacecraft clock information. To extract the information from the returned value, the following masks can be used as in the following: `if ((ReturnValue & CFE_TIME_FLAG_xxxxxx) == CFE_TIME_FLAG_xxxxxx)` then the following definition of the `CFE_TIME_FLAG_xxxxxx` is true.	
CFE_TIME_FLAG_CLKSET	The spacecraft time has been set.
CFE_TIME_FLAG_FLYING	This instance of Time Services is flywheeling.
CFE_TIME_FLAG_SRCINT	The clock source is set to "internal".
CFE_TIME_FLAG_SIGPRI	The clock signal is set to "primary".
CFE_TIME_FLAG_SRVFLY	The Time Server is in flywheel mode.
CFE_TIME_FLAG_CMDFLY	This instance of Time Services was commanded into flywheel mode.
CFE_TIME_FLAG_ADDADJ	One time STCF Adjustment is to be done in positive direction.
CFE_TIME_FLAG_ADD1HZ	1 Hz STCF Adjustment is to be done in a positive direction
CFE_TIME_FLAG_ADDTCL	Time Client Latency is applied in a positive direction.
CFE_TIME_FLAG_SERVER	This instance of Time Services is a Time Server.
~~CFE_TIME_FLAG_GDTONE~~	~~The tone received is good compared to the last tone received.~~
CFE_TIME_FLAG_UNUSED	Reserved flags - should be zero.

Returns

See also

CFE_TIME_GetSTCF, CFE_TIME_GetLeapSeconds, CFE_TIME_GetClockState

13.69.4.10 CFE_TIME_GetClockState()

```
CFE_TIME_ClockState_Enum_t CFE_TIME_GetClockState (
            void  )
```

Description

This routine returns the spacecraft clock state. Applications that are highly dependent on valid time may want to call this routine before taking actions based on the times returned by the various clock routines

Assumptions, External Events, and Notes:

None

CFE_TIME_ClockState_Enum_t	The current spacecraft clock state

Returns

See also

CFE_TIME_GetSTCF, CFE_TIME_GetLeapSeconds, CFE_TIME_GetClockInfo

13.69.4.11 CFE_TIME_GetLeapSeconds()

```
int16 CFE_TIME_GetLeapSeconds (
            void  )
```

Description

This routine returns the current value of the leap seconds counter. This is the delta seconds between international atomic time (TAI) and universal coordinated time (UTC). Applications cannot set or adjust the leap seconds; that can only be done through ground commands. However, science applications may want to include the leap seconds counter in their data products to aid in time correlation during downstream science data processing. Note that some mission operations teams do not maintain the leap seconds count, preferring to adjust the STCF instead. Users of this function should check with their mission ops team to see how they are planning to handle leap seconds.

Assumptions, External Events, and Notes:

None

The current leap seconds.

Returns

See also

CFE_TIME_GetSTCF, CFE_TIME_GetClockState, CFE_TIME_GetClockInfo

13.69.4.12 CFE_TIME_GetMET()

```
CFE_TIME_SysTime_t CFE_TIME_GetMET (
            void  )
```

Description

This routine returns the current mission-elapsed time (MET). MET is usually derived from a hardware-based clock that is not adjusted during normal operations. Callers of this routine should not assume that the MET return value has any specific relationship to any ground-based time standard.

Assumptions, External Events, and Notes:

None

CFE_TIME_SysTime_t	The current MET

Returns

See also

CFE_TIME_GetTime, CFE_TIME_GetTAI, CFE_TIME_GetUTC, CFE_TIME_GetMETseconds, CFE_TIME_GetMETsubsecs, CFE_TIME_MET2SCTime

13.69.4.13 CFE_TIME_GetMETseconds()

```
uint32 CFE_TIME_GetMETseconds (
            void  )
```

Description

This routine is the same as CFE_TIME_GetMET(), except that it returns only the integer seconds portion of the MET time.

Assumptions, External Events, and Notes:

None

Returns

The current MET seconds

See also

CFE_TIME_GetTime, CFE_TIME_GetTAI, CFE_TIME_GetUTC, CFE_TIME_GetMET, CFE_TIME_GetMETsubsecs, CFE_TIME_MET2SCTime

13.69.4.14 CFE_TIME_GetMETsubsecs()

```
uint32 CFE_TIME_GetMETsubsecs (
            void  )
```

Description

This routine is the same as CFE_TIME_GetMET(), except that it returns only the integer sub-seconds portion of the MET time. Each count is equal to 2^(-32) seconds.

Assumptions, External Events, and Notes:

None

Returns

The current MET sub-seconds

See also

CFE_TIME_GetTime, CFE_TIME_GetTAI, CFE_TIME_GetUTC, CFE_TIME_GetMET, CFE_TIME_GetMETseconds, CFE_TIME_MET2SCTime

13.69.4.15 CFE_TIME_GetSTCF()

```
CFE_TIME_SysTime_t CFE_TIME_GetSTCF (
            void  )
```

Description

This routine returns the current value of the spacecraft time correction factor. This is the delta time between the MET and the TAI time. Applications cannot set or adjust the STCF; that can only be done through ground commands. However, science applications may want to include the STCF in their data products to aid in time correlation during downstream science data processing.

Assumptions, External Events, and Notes:

None

CFE_TIME_SysTime_t	The current STCF

Returns

See also

CFE_TIME_GetLeapSeconds, CFE_TIME_GetClockState, CFE_TIME_GetClockInfo

13.69.4.16 CFE_TIME_GetTAI()

```
CFE_TIME_SysTime_t CFE_TIME_GetTAI (
            void  )
```

Description

This routine returns the current TAI time to the caller. TAI is an international time standard that does not include leap seconds. This routine should only be used in situations where TAI is absolutely required. Applications that call CFE_TIME_GetTAI() may not be portable to all missions. Maintenance of correct TAI in flight is not guaranteed under all mission operations scenarios. To maintain re-usability across missions, most applications should be using CFE_TIME_GetTime(), rather than the specific routines for getting UTC/TAI directly.

Assumptions, External Events, and Notes:

1. The "TAI" time returned is referenced to the mission-defined time epoch, which may or may not be the same as the standard TAI epoch.
2. Even though TAI does not include leap seconds, the time returned by this function can still jump forward or backward without warning when the spacecraft clock is set or adjusted by operators. Applications using this function must be able to handle these time discontinuities gracefully.

CFE_TIME_SysTime_t	The current TAI time

Returns

See also

CFE_TIME_GetTime, CFE_TIME_GetUTC, CFE_TIME_GetMET, CFE_TIME_GetMETseconds, CFE_TIME_GetMETsubsecs

13.69.4.17 CFE_TIME_GetTime()

```
CFE_TIME_SysTime_t CFE_TIME_GetTime (
            void  )
```

Description

This routine returns the current spacecraft time. The time returned is either TAI (no leap seconds) or UTC (including leap seconds). This choice is made in the mission configuration file by defining either CFE_MISSION_TIME_CFG_DEFAULT_TAI or CFE_MISSION_TIME_CFG_DEFAULT_UTC as true at compile time. To maintain re-usability across missions, most applications should be using this function (or CFE_TIME_GetTime) rather than the specific routines for getting UTC/TAI directly.

Assumptions, External Events, and Notes:

None

CFE_TIME_SysTime_t	The current spacecraft time

Returns

See also

CFE_TIME_GetTAI, CFE_TIME_GetUTC, CFE_TIME_GetMET, CFE_TIME_GetMETseconds, CFE_TIME_↩ GetMETsubsecs

Referenced by CFE_ES_SysLog_vsnprintf(), CFE_ES_WriteToERLog(), CFE_EVS_SendEvent(), CFE_EVS_Send↩ EventWithAppID(), CFE_SB_TimeStampMsg(), and EVS_SendEvent().

13.69.4.18 CFE_TIME_GetUTC()

```
CFE_TIME_SysTime_t CFE_TIME_GetUTC (
            void  )
```

Description

This routine returns the current UTC time to the caller. This routine should only be used in situations where UTC is absolutely required. Applications that call CFE_TIME_GetUTC() may not be portable to all missions. Maintenance of correct UTC in flight is not guaranteed under all mission operations scenarios. If UTC is maintained in flight, it will jump backwards occasionally due to leap second adjustments. To maintain re-usability across missions, most applications should be using CFE_TIME_GetTime(), rather than the specific routines for getting UTC/TAI directly.

Assumptions, External Events, and Notes:

Note: The "UTC" time returned is referenced to the mission-defined time epoch, which may or may not be the same as the standard UTC epoch.

CFE_TIME_Sys↩ Time_t	The current UTC time

Returns

See also

CFE_TIME_GetTime, CFE_TIME_GetTAI, CFE_TIME_GetMET, CFE_TIME_GetMETseconds, CFE_TIME_↩ GetMETsubsecs

13.69.4.19 CFE_TIME_Local1HzISR()

```
void CFE_TIME_Local1HzISR (
            void  )
```

Description

Drives the time processing logic from the system PSP layer. This must be called once per second based on a hardware interrupt or OS kernel signal.

Assumptions, External Events, and Notes:

This will update the global data structures accordingly, incrementing each by the 1Hz amount.

13.69.4.20 CFE_TIME_MET2SCTime()

```
CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (
            CFE_TIME_SysTime_t METTime )
```

Description

This function returns Spacecraft Time given MET. Note that Spacecraft Time is returned as either UTC or TAI depeneding on whether the mission configuration parameter CFE_MISSION_TIME_CFG_DEFAULT_UTC or CFE_MISSION_TIME_CFG_DEFAULT_TAI was set to true at compile time.

Assumptions, External Events, and Notes:

None

Parameters

in	*METTime*	The MET to be converted.

CFE_TIME_SysTime_t	Spacecraft Time (UTC or TAI) corresponding to the specified MET

Returns

See also

CFE_TIME_GetMET, CFE_TIME_GetMETseconds, CFE_TIME_GetMETsubsecs, CFE_TIME_Sub2MicroSecs, CFE_TIME_Micro2SubSecs, CFE_TIME_CFE2FSSeconds, CFE_TIME_FS2CFESeconds

13.69.4.21 CFE_TIME_Micro2SubSecs()

```
uint32 CFE_TIME_Micro2SubSecs (
            uint32 MicroSeconds )
```

Description

This routine converts from microseconds (each tick is 1e-06 seconds) to a subseconds count (each tick is 1 / 2^32 seconds).

Assumptions, External Events, and Notes:

None

Parameters

in	*MicroSeconds*	The sub-seconds count to convert.

The equivalent number of subseconds. If the number of microseconds passed in is greater than one second, (i.e. > 999,999), the return value is equal to `0xffffffff`.

Returns

See also

CFE_TIME_MET2SCTime, CFE_TIME_Sub2MicroSecs, CFE_TIME_CFE2FSSeconds, CFE_TIME_FS2CFE↩ Seconds

Referenced by CFE_SB_GetMsgTime().

13.69.4.22 CFE_TIME_Print()

```
void CFE_TIME_Print (
            char * PrintBuffer,
            CFE_TIME_SysTime_t TimeToPrint )
```

Description

This routine prints the specified time to the specified string buffer in the following format:

`yyyy-ddd-hh`:mm:ss.xxxxx\0

where:

- `yyyy` = year
- `ddd` = Julian day of the year
- `hh` = hour of the day (0 to 23)
- `mm` = minute (0 to 59)
- `ss` = second (0 to 59)
- `xxxxx` = subsecond formatted as a decimal fraction (1/4 second = 0.25000)
- `\0` = trailing null

Assumptions, External Events, and Notes:

None

Parameters

in	*PrintBuffer*	Pointer to a character array of at least CFE_TIME_PRINTED_STRING_SIZE characters in length
in	*TimeToPrint*	The time to print into the character array.
out	∗*PrintBuffer*	The time as a character string as described above.

See also

Referenced by CFE_ES_SysLog_vsnprintf().

13.69.4.23 CFE_TIME_RegisterSynchCallback()

```
int32 CFE_TIME_RegisterSynchCallback (
            CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr )
```

Description

This routine passes a callback function pointer for an Application that wishes to be notified whenever a legitimate time synchronization signal (typically a 1 Hz) is received.

Assumptions, External Events, and Notes:

Only a single callback per application is supported, and this function should only be called from a single thread within each application (typically the apps main thread). If an application requires triggering multiple child tasks at 1Hz, it should distribute the timing signal internally, rather than registering for multiple callbacks.

CFE_SUCCESS	Operation was performed successfully
CFE_TIME_TOO_MANY_SYNCH_CALLBACKS	An attempt to register too many cFE Time Services Synchronization callbacks has been made. Only one callback function is allowed per application. It is expected that the application itself will distribute the single callback to child threads as needed.
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.

Returns

See also

CFE_TIME_UnregisterSynchCallback

13.69.4.24 CFE_TIME_Sub2MicroSecs()

```
uint32 CFE_TIME_Sub2MicroSecs (
            uint32 SubSeconds )
```

Description

This routine converts from a sub-seconds count (each tick is 1 / 2^32 seconds) to microseconds (each tick is 1e-06 seconds).

Assumptions, External Events, and Notes:

None

Parameters

in	*SubSeconds*	The sub-seconds count to convert.

Returns

The equivalent number of microseconds.

See also

CFE_TIME_MET2SCTime, CFE_TIME_Micro2SubSecs, CFE_TIME_CFE2FSSeconds, CFE_TIME_FS2CFE↩ Seconds

Referenced by CFE_SB_SetMsgTime().

13.69.4.25 CFE_TIME_Subtract()

```
CFE_TIME_SysTime_t CFE_TIME_Subtract (
            CFE_TIME_SysTime_t Time1,
            CFE_TIME_SysTime_t Time2 )
```

Description

This routine subtracts time2 from time1 and returns the result. The time values can represent either absolute or delta times, but not all combinations make sense.

- AbsTime - AbsTime = DeltaTime
- AbsTime - DeltaTime = AbsTime

- DeltaTime - DeltaTime = DeltaTime
- DeltaTime - AbsTime = garbage

Assumptions, External Events, and Notes:

None

Parameters

in	*Time1*	The base time.
in	*Time2*	The time to be subtracted from the base time.

The result of subtracting the two times, in the CFE_TIME_SysTime_t format. If the subtraction results in an underflow, the result will roll over (this is not considered an error).

Returns

See also

CFE_TIME_Add, CFE_TIME_Compare

13.69.4.26 CFE_TIME_UnregisterSynchCallback()

```
int32 CFE_TIME_UnregisterSynchCallback (
            CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr )
```

Description

This routine removes the specified callback function pointer from the list of Callback functions that are called whenever a time synchronization (typically the 1Hz signal) is received.

Assumptions, External Events, and Notes:

Only a single callback per application is supported, and this function should only be called from a single thread within each application (typically the apps main thread).

CFE_SUCCESS	Operation was performed successfully
CFE_TIME_CALLBACK_NOT_REGISTERED	An attempt to unregister a cFE Time Services Synchronization callback has failed because the specified callback function was not located in the Synchronization Callback Registry.
CFE_ES_ERR_APPID	The given application ID does not reflect a currently active application.

Returns

See also

CFE_TIME_RegisterSynchCallback

13.70 cfe/fsw/cfe-core/src/inc/cfe_time_events.h File Reference

Macros

- #define CFE_TIME_MAX_EID 49
- #define CFE_TIME_INIT_EID 1 /* start up message "informational" */

  'cFE TIME Initialized'
- #define CFE_TIME_NOOP_EID 4 /* processed command "informational" */

  'No-op command'
- #define CFE_TIME_RESET_EID 5

  'Reset Counters command'
- #define CFE_TIME_DIAG_EID 6

  'Request diagnostics command'
- #define CFE_TIME_STATE_EID 7

  'Set Clock State = %s'
- #define CFE_TIME_SOURCE_EID 8

  'Set Time Source = %s'
- #define CFE_TIME_SIGNAL_EID 9

  'Set Tone Source = %s'
- #define CFE_TIME_DELAY_EID 11

  'Set Tone Delay - secs = %d, usecs = %d, ssecs = 0x%X, dir = %d'
- #define CFE_TIME_TIME_EID 12

  'Set Time - secs = %d, usecs = %d, ssecs = 0x%X'
- #define CFE_TIME_MET_EID 13

  'Set MET - secs = %d, usecs = %d, ssecs = 0x%X'
- #define CFE_TIME_STCF_EID 14

  'Set STCF - secs = %d, usecs = %d, ssecs = 0x%X'
- #define CFE_TIME_DELTA_EID 15

  'STCF Adjust - secs = %d, usecs = %d, ssecs = 0x%X, dir[1=Positive, 2=Negative] = %d'
- #define CFE_TIME_1HZ_EID 16

  'STCF 1Hz Adjust - secs = %d, ssecs = 0x%X, dir = %d'
- #define CFE_TIME_LEAPS_EID 17

  'Set Leap Seconds = %d'
- #define CFE_TIME_FLY_ON_EID 20 /* flywheel state "informational" */

  'Start FLYWHEEL'
- #define CFE_TIME_FLY_OFF_EID 21

  'Stop FLYWHEEL'
- #define CFE_TIME_EXIT_ERR_EID 25 /* task termination "error" */

- #define CFE_TIME_ID_ERR_EID 26 /* invalid command packet "error" */

  'Invalid message ID - ID = 0x%X'
- #define CFE_TIME_CC_ERR_EID 27

  'Invalid command code - ID = 0x%X, CC = %d'
- #define CFE_TIME_STATE_ERR_EID 30 /* processed command "error" */

  'Invalid Clock State = 0x%X'
- #define CFE_TIME_SOURCE_ERR_EID 31

  'Invalid Time Source = 0x%X'
- #define CFE_TIME_SIGNAL_ERR_EID 32

  'Invalid Tone Source = 0x%X'
- #define CFE_TIME_DELAY_ERR_EID 33

  'Invalid Tone Delay - secs = %d, usecs = %d'
- #define CFE_TIME_TIME_ERR_EID 34

  'Invalid Time - secs = %d, usecs = %d'
- #define CFE_TIME_MET_ERR_EID 35

  'Invalid MET - secs = %d, usecs = %d'
- #define CFE_TIME_STCF_ERR_EID 36

  'Invalid STCF - secs = %d, usecs = %d'
- #define CFE_TIME_DELTA_ERR_EID 37

  'Invalid STCF Adjust - secs = %d, usecs = %d, dir[1=Positive, 2=Negative] = %d'
- #define CFE_TIME_1HZ_ERR_EID 38
- #define CFE_TIME_SOURCE_CFG_EID 40 /* cmd disabled per cfg "error" */

  'Set Source commands invalid without CFE_PLATFORM_TIME_CFG_SOURCE set to true'
- #define CFE_TIME_SIGNAL_CFG_EID 41

  'Set Signal commands invalid without CFE_PLATFORM_TIME_CFG_SIGNAL set to true'
- #define CFE_TIME_DELAY_CFG_EID 42

  'Set Delay commands invalid without CFE_PLATFORM_TIME_CFG_CLIENT set to true'
- #define CFE_TIME_TIME_CFG_EID 43

  'Set Time commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
- #define CFE_TIME_MET_CFG_EID 44

  'Set MET commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
- #define CFE_TIME_STCF_CFG_EID 45

  'Set STCF commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
- #define CFE_TIME_LEAPS_CFG_EID 46

  'Set Leaps commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
- #define CFE_TIME_DELTA_CFG_EID 47

  'STCF Adjust commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
- #define CFE_TIME_1HZ_CFG_EID 48

  '1Hz Adjust commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'
- #define CFE_TIME_LEN_ERR_EID 49

  'Invalid cmd length: ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'

13.70.1 Macro Definition Documentation

13.70.1.1 CFE_TIME_1HZ_CFG_EID

```
#define CFE_TIME_1HZ_CFG_EID 48
```

Event Message `'1Hz Adjust commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives either a Add STCF Adjustment each second Command OR a Subtract STCF Adjustment each second command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SERVER has not been set to true in the cfe_platform_cfg.h file.

Definition at line 603 of file cfe_time_events.h.

13.70.1.2 CFE_TIME_1HZ_EID

```
#define CFE_TIME_1HZ_EID 16
```

Event Message `'STCF 1Hz Adjust - secs = %d, ssecs = 0x%X, dir = %d'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of any of the following cFE Time Services STCF Adjustment Commands:

- Add STCF Adjustment each second command
- Subtract STCF Adjustment each second command

The `secs` field specifies the number of seconds the STCF is to be adjusted by, the `ssecs` field specifies the number of sub-seconds (1/2^32 seconds) the STCF is to be adjusted by and the `dir` field identifies whether the adjustment was added or subtracted. The direction value can be either CFE_TIME_AdjustDirection_ADD or CFE_TIME_AdjustDirection_SUBTRACT.

Definition at line 251 of file cfe_time_events.h.

13.70.1.3 CFE_TIME_1HZ_ERR_EID

```
#define CFE_TIME_1HZ_ERR_EID 38
```

(obsolete - unused)

Definition at line 474 of file cfe_time_events.h.

13.70.1.4 CFE_TIME_CC_ERR_EID

```
#define CFE_TIME_CC_ERR_EID 27
```

Event Message `'Invalid command code - ID = 0x%X, CC = %d'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a message from the software bus that contains a unrecognized command code in its header..

The `ID` field specifies, in hex, the message ID of the message containing the unrecognized command code, identified, in decimal, by the `CC` field.

Definition at line 322 of file cfe_time_events.h.

13.70.1.5 CFE_TIME_DELAY_CFG_EID

```
#define CFE_TIME_DELAY_CFG_EID 42
```

Event Message `'Set Delay commands invalid without CFE_PLATFORM_TIME_CFG_CLIENT set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives either a Add Tone Delay Command OR a Subtract Tone Delay Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_CLIENT has not been set to true in the cfe_platform_cfg.h file.

Definition at line 517 of file cfe_time_events.h.

13.70.1.6 CFE_TIME_DELAY_EID

```
#define CFE_TIME_DELAY_EID 11
```

Event Message `'Set Tone Delay - secs = %d, usecs = %d, ssecs = 0x%X, dir = %d'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of either a cFE Time Services Add Time Delay OR a Subtract Time Delay command

The `secs` field specifies the new delay (in seconds), the `usecs` field specifies the delay in micro-seconds, the `ssecs` field is the micro-seconds field converted to Spacecraft Time sub-seconds and the `dir` field identifies the direction of the delay. The direction can be either CFE_TIME_AdjustDirection_ADD or CFE_TIME_AdjustDirection_SUBTRACT.

Definition at line 162 of file cfe_time_events.h.

13.70.1.7 CFE_TIME_DELAY_ERR_EID

```
#define CFE_TIME_DELAY_ERR_EID 33
```

Event Message `'Invalid Tone Delay - secs = %d, usecs = %d'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives either a Add Tone Delay Command OR a Subtract Tone Delay Command that contains a microsecond field that is greater than or equal to 1000000.

The `secs` field specifies, in decimal, the tone signal delay in seconds and the `usecs` field specifies, in decimal, the micro-second delay that was in error.

Definition at line 396 of file cfe_time_events.h.

13.70.1.8 CFE_TIME_DELTA_CFG_EID

```
#define CFE_TIME_DELTA_CFG_EID 47
```

Event Message `'STCF Adjust commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives either a Add Single STCF Adjustment Command OR a Subtract Single STCF Adjustment command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SERVER has not been set to true in the cfe_platform_cfg.h file.

Definition at line 588 of file cfe_time_events.h.

13.70.1.9 CFE_TIME_DELTA_EID

```
#define CFE_TIME_DELTA_EID 15
```

Event Message `'STCF Adjust - secs = %d, usecs = %d, ssecs = 0x%X, dir[1=Positive, 2=Negative] = %d'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of any of the following cFE Time Services STCF Adjustment Commands:

- Add Single STCF Adjustment command
- Subtract Single STCF Adjustment command

The `secs` field specifies the number of seconds the STCF is to be adjusted by, the `usecs` field specifies the number of micro-seconds, the `ssecs` field is the micro-seconds field converted to Spacecraft Time sub-seconds and the `dir` field identifies whether the adjustment was added or subtracted. The direction can be either CFE_TIME_AdjustDirection_ADD or CFE_TIME_AdjustDirection_SUBTRACT.

Definition at line 231 of file cfe_time_events.h.

13.70.1.10 CFE_TIME_DELTA_ERR_EID

```
#define CFE_TIME_DELTA_ERR_EID 37
```

Event Message `'Invalid STCF Adjust - secs = %d, usecs = %d, dir[1=Positive, 2=Negative] = %d'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives either a Add Single STCF Adjustment Command OR a Subtract Single STCF Adjustment command that contains a microsecond field that is greater than or equal to 1,000,000.

The `secs` field specifies the number of seconds the STCF is to be adjusted by, the `usecs` field specifies the number of micro-seconds that was in error, the `dir` field identifies whether the adjustment was to be added or subtracted. The direction can be either CFE_TIME_AdjustDirection_ADD or CFE_TIME_AdjustDirection_SUBTRACT.

Definition at line 470 of file cfe_time_events.h.

13.70.1.11 CFE_TIME_DIAG_EID

```
#define CFE_TIME_DIAG_EID 6
```

Event Message `'Request diagnostics command'`

Type: DEBUG

Cause:

This event message is always automatically issued in response to a cFE Time Services Request Diagnostics command

Definition at line 96 of file cfe_time_events.h.

13.70.1.12 CFE_TIME_EXIT_ERR_EID

```
#define CFE_TIME_EXIT_ERR_EID 25 /* task termination "error" */
```

Definition at line 290 of file cfe_time_events.h.

13.70.1.13 CFE_TIME_FLY_OFF_EID

```
#define CFE_TIME_FLY_OFF_EID 21
```

Event Message `'Stop FLYWHEEL'`

Type: INFORMATION

Cause:

This event message is generated whenever the Time Services exits FLYWHEEL mode.

Definition at line 288 of file cfe_time_events.h.

13.70.1.14 CFE_TIME_FLY_ON_EID

```
#define CFE_TIME_FLY_ON_EID 20 /* flywheel state "informational" */
```

Event Message `'Start FLYWHEEL'`

Type: INFORMATION

Cause:

This event message is generated whenever the Time Services enters FLYWHEEL mode.

Definition at line 277 of file cfe_time_events.h.

13.70.1.15 CFE_TIME_ID_ERR_EID

```
#define CFE_TIME_ID_ERR_EID 26 /* invalid command packet "error" */
```

Event Message `'Invalid message ID - ID = 0x%X'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a message from the software bus that is not one of Time Services recognized messages.

The `ID` field specifies, in hex, the message ID of the inappropriately received message.

Definition at line 306 of file cfe_time_events.h.

13.70.1.16 CFE_TIME_INIT_EID

```
#define CFE_TIME_INIT_EID 1 /* start up message "informational" */
```

Event Message `'cFE TIME Initialized'`

Type: INFORMATION

Cause:

This event message is always automatically issued when the Time Services Task completes its Initialization.

Definition at line 60 of file cfe_time_events.h.

13.70.1.17 CFE_TIME_LEAPS_CFG_EID

```
#define CFE_TIME_LEAPS_CFG_EID 46
```

Event Message `'Set Leaps commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Leap Seconds Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SERVER has not been set to true in the cfe_platform_↩cfg.h file.

Definition at line 573 of file cfe_time_events.h.

13.70.1.18 CFE_TIME_LEAPS_EID

```
#define CFE_TIME_LEAPS_EID 17
```

Event Message `'Set Leap Seconds = %d'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of the Set Leap Seconds command

The `%d` field contains the number of seconds the Spacecraft's Leap Seconds has been set to.

Definition at line 266 of file cfe_time_events.h.

13.70.1.19 CFE_TIME_LEN_ERR_EID

```
#define CFE_TIME_LEN_ERR_EID 49
```

Event Message `'Invalid cmd length:  ID = 0x%X, CC = %d, Exp Len = %d, Len = %d'`

Type: ERROR

Cause:

This event message is generated when a message with the CFE_TIME_CMD_MID message ID has arrived but whose packet length does not match the expected length for the specified command code.

The `ID` field in the event message specifies the Message ID (in hex), the `CC` field specifies the Command Code (in decimal), the `Exp` Len field specified the Expected Length (in decimal), and `Len` specifies the message Length (in decimal) found in the message.

Definition at line 621 of file cfe_time_events.h.

13.70.1.20 CFE_TIME_MAX_EID

```
#define CFE_TIME_MAX_EID 49
```

Definition at line 45 of file cfe_time_events.h.

13.70.1.21 CFE_TIME_MET_CFG_EID

```
#define CFE_TIME_MET_CFG_EID 44
```

Event Message `'Set MET commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Mission Elapsed Time Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SERVER has not been set to true in the cfe_↩ platform_cfg.h file.

Definition at line 545 of file cfe_time_events.h.

13.70.1.22 CFE_TIME_MET_EID

```
#define CFE_TIME_MET_EID 13
```

Event Message `'Set MET - secs = %d, usecs = %d, ssecs = 0x%X'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of a cFE Time Services Set Mission Elapsed Time command

The `secs` field specifies the new MET (in seconds), the `usecs` field specifies the MET micro-seconds, the `ssecs` field is the micro-seconds field converted to Spacecraft Time sub-seconds

Definition at line 194 of file cfe_time_events.h.

13.70.1.23 CFE_TIME_MET_ERR_EID

```
#define CFE_TIME_MET_ERR_EID 35
```

Event Message `'Invalid MET - secs = %d, usecs = %d'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Mission Elapsed Time Command that contains a microsecond field that is greater than or equal to 1,000,000.

The `secs` field specifies, in decimal, the MET in seconds and the `usecs` field specifies, in decimal, the micro-second field of the MET that was in error.

Definition at line 432 of file cfe_time_events.h.

13.70.1.24 CFE_TIME_NOOP_EID

```
#define CFE_TIME_NOOP_EID 4 /* processed command "informational" */
```

Event Message 'No-op command'

Type: INFORMATION

Cause:

This event message is always automatically issued in response to a cFE Time Services NO-OP command

Definition at line 72 of file cfe_time_events.h.

13.70.1.25 CFE_TIME_RESET_EID

```
#define CFE_TIME_RESET_EID 5
```

Event Message 'Reset Counters command'

Type: DEBUG

Cause:

This event message is always automatically issued in response to a cFE Time Services Reset Counters command

Definition at line 84 of file cfe_time_events.h.

13.70.1.26 CFE_TIME_SIGNAL_CFG_EID

```
#define CFE_TIME_SIGNAL_CFG_EID 41
```

Event Message `'Set Signal commands invalid without CFE_PLATFORM_TIME_CFG_SIGNAL set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Clock Signal Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SIGNAL has not been set to true in the cfe_platform_↩ cfg.h file.

Definition at line 502 of file cfe_time_events.h.

13.70.1.27 CFE_TIME_SIGNAL_EID

```
#define CFE_TIME_SIGNAL_EID 9
```

Event Message `'Set Tone Source = %s'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of a cFE Time Services Set Clock Signal command

The `'%s'` field will identify whether the command specified `PRIMARY`, or `REDUNDANT`.

Definition at line 141 of file cfe_time_events.h.

13.70.1.28 CFE_TIME_SIGNAL_ERR_EID

```
#define CFE_TIME_SIGNAL_ERR_EID 32
```

Event Message `'Invalid Tone Source = 0x%X'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Clock Signal Command that contains a desired clock source that is none of the following:

- CFE_TIME_ToneSignalSelect_PRIMARY
- CFE_TIME_ToneSignalSelect_REDUNDANT

The `Source` field specifies, in hex, the signal source value received in the command message.

Definition at line 377 of file cfe_time_events.h.

13.70.1.29 CFE_TIME_SOURCE_CFG_EID

```
#define CFE_TIME_SOURCE_CFG_EID 40 /* cmd disabled per cfg "error" */
```

Event Message `'Set Source commands invalid without CFE_PLATFORM_TIME_CFG_SOURCE set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Clock Source Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SOURCE has not been set to true in the cfe_platform_↩ cfg.h file.

Definition at line 488 of file cfe_time_events.h.

13.70.1.30 CFE_TIME_SOURCE_EID

```
#define CFE_TIME_SOURCE_EID 8
```

Event Message `'Set Time Source = %s'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of a cFE Time Services Set Time Source command

The `'%s'` field will identify whether the command specified `INTERNAL`, or `EXTERNAL`.

Definition at line 126 of file cfe_time_events.h.

13.70.1.31 CFE_TIME_SOURCE_ERR_EID

```
#define CFE_TIME_SOURCE_ERR_EID 31
```

Event Message `'Invalid Time Source = 0x%X'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Clock Source Command that contains a desired clock source that is none of the following:

- CFE_TIME_SourceSelect_INTERNAL
- CFE_TIME_SourceSelect_EXTERNAL

The `Source` field specifies, in hex, the source value received in the command message.

Definition at line 359 of file cfe_time_events.h.

13.70.1.32 CFE_TIME_STATE_EID

```
#define CFE_TIME_STATE_EID 7
```

Event Message `'Set Clock State = %s'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of a cFE Time Services Set Time State command

The `'%s'` field will identify whether the command specified `VALID`, `INVALID`, or `FLYWHEEL`.

Definition at line 111 of file cfe_time_events.h.

13.70.1.33 CFE_TIME_STATE_ERR_EID

```
#define CFE_TIME_STATE_ERR_EID 30 /* processed command "error" */
```

Event Message `'Invalid Clock State = 0x%X'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Clock State Command that contains a desired clock state that is none of the following:

- CFE_TIME_ClockState_INVALID
- CFE_TIME_ClockState_VALID
- CFE_TIME_ClockState_FLYWHEEL

The `State` field specifies, in hex, the state value received in the command message.

Definition at line 341 of file cfe_time_events.h.

13.70.1.34 CFE_TIME_STCF_CFG_EID

```
#define CFE_TIME_STCF_CFG_EID 45
```

Event Message `'Set STCF commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Spacecraft Time Correlation Factor Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SERVER has not been set to true in the cfe_platform_cfg.h file.

Definition at line 559 of file cfe_time_events.h.

13.70.1.35 CFE_TIME_STCF_EID

```
#define CFE_TIME_STCF_EID 14
```

Event Message `'Set STCF - secs = %d, usecs = %d, ssecs = 0x%X'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of a cFE Time Services Set Spacecraft Time Correlation Factor command

The `secs` field specifies the new STCF (in seconds), the `usecs` field specifies the STCF micro-seconds, the `ssecs` field is the micro-seconds field converted to Spacecraft Time sub-seconds.

Definition at line 211 of file cfe_time_events.h.

13.70.1.36 CFE_TIME_STCF_ERR_EID

```
#define CFE_TIME_STCF_ERR_EID 36
```

Event Message `'Invalid STCF - secs = %d, usecs = %d'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Spacecraft Time Correlation Factor Command that contains a microsecond field that is greater than or equal to 1,000,000.

The `secs` field specifies, in decimal, the STCF in seconds and the `usecs` field specifies, in decimal, the micro-second field of the STCF that was in error.

Definition at line 450 of file cfe_time_events.h.

13.70.1.37 CFE_TIME_TIME_CFG_EID

```
#define CFE_TIME_TIME_CFG_EID 43
```

Event Message `'Set Time commands invalid without CFE_PLATFORM_TIME_CFG_SERVER set to true'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Spacecraft Time Command and the Time Services configuration parameter CFE_PLATFORM_TIME_CFG_SERVER has not been set to true in the cfe_platform_↩cfg.h file.

Definition at line 531 of file cfe_time_events.h.

13.70.1.38 CFE_TIME_TIME_EID

```
#define CFE_TIME_TIME_EID 12
```

Event Message `'Set Time - secs = %d, usecs = %d, ssecs = 0x%X'`

Type: INFORMATION

Cause:

This event message is generated upon successful completion of a cFE Time Services Set Time command

The `secs` field specifies the new spacecraft time (in seconds), the `usecs` field specifies the spacecraft time micro-seconds, the `ssecs` field is the micro-seconds field converted to Spacecraft Time sub-seconds

Definition at line 178 of file cfe_time_events.h.

13.70.1.39 CFE_TIME_TIME_ERR_EID

```
#define CFE_TIME_TIME_ERR_EID 34
```

Event Message `'Invalid Time - secs = %d, usecs = %d'`

Type: ERROR

Cause:

This event message is generated whenever Time Services receives a Set Spacecraft Time Command that contains a microsecond field that is greater than or equal to 1,000,000.

The `secs` field specifies, in decimal, the spacecraft time in seconds and the `usecs` field specifies, in decimal, the micro-second field of the spacecraft time that was in error.

Definition at line 414 of file cfe_time_events.h.

13.71 cfe/fsw/cfe-core/src/inc/cfe_time_extern_typedefs.h File Reference

```
#include "common_types.h"
```

Typedefs

- typedef uint8 CFE_TIME_FlagBit_Enum_t

  Bit positions of the various clock state flags.
- typedef int16 CFE_TIME_ClockState_Enum_t

  Enumerated types identifying the quality of the current time.
- typedef uint8 CFE_TIME_SourceSelect_Enum_t

  Clock Source Selection Parameters.
- typedef uint8 CFE_TIME_ToneSignalSelect_Enum_t

  Tone Signal Selection Parameters.
- typedef uint8 CFE_TIME_AdjustDirection_Enum_t

  STCF adjustment direction (for both one-time and 1Hz adjustments)
- typedef uint8 CFE_TIME_FlywheelState_Enum_t

  Fly-wheel status values.
- typedef uint8 CFE_TIME_SetState_Enum_t

  Clock status values (has the clock been set to correct time)

Enumerations

- enum CFE_TIME_FlagBit {
  CFE_TIME_FlagBit_CLKSET = 0, CFE_TIME_FlagBit_FLYING = 1, CFE_TIME_FlagBit_SRCINT = 2, CFE_T↩
  IME_FlagBit_SIGPRI = 3,
  CFE_TIME_FlagBit_SRVFLY = 4, CFE_TIME_FlagBit_CMDFLY = 5, CFE_TIME_FlagBit_ADDADJ = 6, CFE_↩
  TIME_FlagBit_ADD1HZ = 7,
  CFE_TIME_FlagBit_ADDTCL = 8, CFE_TIME_FlagBit_SERVER = 9, CFE_TIME_FlagBit_GDTONE = 10 }

  Label definitions associated with CFE_TIME_FlagBit_Enum_t.
- enum CFE_TIME_ClockState { CFE_TIME_ClockState_INVALID = -1, CFE_TIME_ClockState_VALID = 0, CF↩
  E_TIME_ClockState_FLYWHEEL = 1 }

  Label definitions associated with CFE_TIME_ClockState_Enum_t.
- enum CFE_TIME_SourceSelect { CFE_TIME_SourceSelect_INTERNAL = 1, CFE_TIME_SourceSelect_EXT↩
  ERNAL = 2 }

  Label definitions associated with CFE_TIME_SourceSelect_Enum_t.
- enum CFE_TIME_ToneSignalSelect { CFE_TIME_ToneSignalSelect_PRIMARY = 1, CFE_TIME_ToneSignal↩
  Select_REDUNDANT = 2 }

  Label definitions associated with CFE_TIME_ToneSignalSelect_Enum_t.
- enum CFE_TIME_AdjustDirection { CFE_TIME_AdjustDirection_ADD = 1, CFE_TIME_AdjustDirection_SUBT↩
  RACT = 2 }

  Label definitions associated with CFE_TIME_AdjustDirection_Enum_t.
- enum CFE_TIME_FlywheelState { CFE_TIME_FlywheelState_NO_FLY = 0, CFE_TIME_FlywheelState_IS_FLY
  = 1 }

  Label definitions associated with CFE_TIME_FlywheelState_Enum_t.
- enum CFE_TIME_SetState { CFE_TIME_SetState_NOT_SET = 0, CFE_TIME_SetState_WAS_SET = 1 }

  Label definitions associated with CFE_TIME_SetState_Enum_t.

13.71.1 Typedef Documentation

13.71.1.1 CFE_TIME_AdjustDirection_Enum_t

```
typedef uint8 CFE_TIME_AdjustDirection_Enum_t
```

See also

enum CFE_TIME_AdjustDirection

Definition at line 237 of file cfe_time_extern_typedefs.h.

13.71.1.2 CFE_TIME_ClockState_Enum_t

```
typedef int16 CFE_TIME_ClockState_Enum_t
```

Description

The CFE_TIME_ClockState_Enum_t enumerations identify the three recognized states of the current time. If the clock has never been successfully synchronized with the primary onboard clock source, the time is conisdered to be CFE_TIME_ClockState_INVALID. If the time is currently synchronized (i.e. - the primary synchronization mechanism has not been dropped for any significant amount of time), then the current time is considered to be CFE_TIME_ClockState_VALID. If the time had, at some point in the past, been synchronized, but the synchronization with the primary onboard clock has since been lost, then the time is considered to be CFE_TIME_ClockState_FLYWHEEL. Since different clocks drift at different rates from one another, the accuracy of the time while in CFE_TIME_ClockState_FLYWHEEL is dependent upon the time spent in that state.

See also

enum CFE_TIME_ClockState

Definition at line 159 of file cfe_time_extern_typedefs.h.

13.71.1.3 CFE_TIME_FlagBit_Enum_t

```
typedef uint8 CFE_TIME_FlagBit_Enum_t
```

See also

enum CFE_TIME_FlagBit

Definition at line 104 of file cfe_time_extern_typedefs.h.

13.71.1.4 CFE_TIME_FlywheelState_Enum_t

```
typedef uint8 CFE_TIME_FlywheelState_Enum_t
```

See also

enum CFE_TIME_FlywheelState

Definition at line 263 of file cfe_time_extern_typedefs.h.

13.71.1.5 CFE_TIME_SetState_Enum_t

```
typedef uint8 CFE_TIME_SetState_Enum_t
```

See also

enum CFE_TIME_SetState

Definition at line 289 of file cfe_time_extern_typedefs.h.

13.71.1.6 CFE_TIME_SourceSelect_Enum_t

```
typedef uint8 CFE_TIME_SourceSelect_Enum_t
```

See also

enum CFE_TIME_SourceSelect

Definition at line 185 of file cfe_time_extern_typedefs.h.

13.71.1.7 CFE_TIME_ToneSignalSelect_Enum_t

```
typedef uint8 CFE_TIME_ToneSignalSelect_Enum_t
```

See also

enum CFE_TIME_ToneSignalSelect

Definition at line 211 of file cfe_time_extern_typedefs.h.

13.71.2 Enumeration Type Documentation

13.71.2.1 CFE_TIME_AdjustDirection

```
enum CFE_TIME_AdjustDirection
```

Enumerator

CFE_TIME_AdjustDirection_ADD	Add time adjustment.
CFE_TIME_AdjustDirection_SUBTRACT	Subtract time adjustment.

Definition at line 217 of file cfe_time_extern_typedefs.h.

13.71.2.2 CFE_TIME_ClockState

`enum CFE_TIME_ClockState`

Enumerator

CFE_TIME_ClockState_INVALID	The spacecraft time has not been set since the last clock reset. Times returned by clock routines have no relationship to any ground-based time reference.
CFE_TIME_ClockState_VALID	The spacecraft time has been set at least once since the last clock reset, and it is synchronized with the primary on-board time base. Times returned by clock routines can be trusted.
CFE_TIME_ClockState_FLYWHEEL	The spacecraft time has been set at least once since the last clock reset, but it is not currently synchronized with the primary on-board time base. Times returned by clock routines are a "best guess" based on a non-optimal oscillator.

Definition at line 110 of file cfe_time_extern_typedefs.h.

13.71.2.3 CFE_TIME_FlagBit

`enum CFE_TIME_FlagBit`

Enumerator

CFE_TIME_FlagBit_CLKSET	The spacecraft time has been set.
CFE_TIME_FlagBit_FLYING	This instance of Time Services is flywheeling.
CFE_TIME_FlagBit_SRCINT	The clock source is set to internal.
CFE_TIME_FlagBit_SIGPRI	The clock signal is set to primary.
CFE_TIME_FlagBit_SRVFLY	The Time Server is in flywheel mode.
CFE_TIME_FlagBit_CMDFLY	This instance of Time Services was commanded into flywheel mode.
CFE_TIME_FlagBit_ADDADJ	One time STCF Adjustment is to be done in positive direction.
CFE_TIME_FlagBit_ADD1HZ	1 Hz STCF Adjustment is to be done in a positive direction
CFE_TIME_FlagBit_ADDTCL	Time Client Latency is applied in a positive direction.
CFE_TIME_FlagBit_SERVER	This instance of Time Services is a Time Server.
CFE_TIME_FlagBit_GDTONE	The tone received is good compared to the last tone received.

Definition at line 39 of file cfe_time_extern_typedefs.h.

13.71.2.4 CFE_TIME_FlywheelState

enum CFE_TIME_FlywheelState

Enumerator

CFE_TIME_FlywheelState_NO_FLY	Not in flywheel state.
CFE_TIME_FlywheelState_IS_FLY	In flywheel state.

Definition at line 243 of file cfe_time_extern_typedefs.h.

13.71.2.5 CFE_TIME_SetState

enum CFE_TIME_SetState

Enumerator

CFE_TIME_SetState_NOT_SET	Spacecraft time has not been set.
CFE_TIME_SetState_WAS_SET	Spacecraft time has been set.

Definition at line 269 of file cfe_time_extern_typedefs.h.

13.71.2.6 CFE_TIME_SourceSelect

enum CFE_TIME_SourceSelect

Enumerator

CFE_TIME_SourceSelect_INTERNAL	Use Internal Source.
CFE_TIME_SourceSelect_EXTERNAL	Use External Source.

Definition at line 165 of file cfe_time_extern_typedefs.h.

13.71.2.7 CFE_TIME_ToneSignalSelect

enum CFE_TIME_ToneSignalSelect

Enumerator

CFE_TIME_ToneSignalSelect_PRIMARY	Primary Source.
CFE_TIME_ToneSignalSelect_REDUNDANT	Redundant Source.

Definition at line 191 of file cfe_time_extern_typedefs.h.

13.72 cfe/fsw/cfe-core/src/inc/cfe_time_msg.h File Reference

```
#include "cfe.h"
```

Data Structures

- struct CFE_TIME_NoArgsCmd_t
- struct CFE_TIME_LeapsCmd_Payload_t
- struct CFE_TIME_SetLeapSeconds_t
- struct CFE_TIME_StateCmd_Payload_t
- struct CFE_TIME_SetState_t
- struct CFE_TIME_SourceCmd_Payload_t
- struct CFE_TIME_SetSource_t
- struct CFE_TIME_SignalCmd_Payload_t
- struct CFE_TIME_SetSignal_t
- struct CFE_TIME_TimeCmd_Payload_t
- struct CFE_TIME_TimeCmd_t
- struct CFE_TIME_OneHzAdjustmentCmd_Payload_t
- struct CFE_TIME_OneHzAdjustmentCmd_t
- struct CFE_TIME_1HzCmd_t
- struct CFE_TIME_ToneSignalCmd_t
- struct CFE_TIME_FakeToneCmd_t
- struct CFE_TIME_ToneDataCmd_Payload_t
- struct CFE_TIME_ToneDataCmd_t
- struct CFE_TIME_HousekeepingTlm_Payload_t
- struct CFE_TIME_HousekeepingTlm_t
- struct CFE_TIME_DiagnosticTlm_Payload_t
- struct CFE_TIME_DiagnosticTlm_t

Macros

Time Services Command Codes

- #define CFE_TIME_NOOP_CC 0 /∗ no-op command ∗/
- #define CFE_TIME_RESET_COUNTERS_CC 1 /∗ reset counters ∗/
- #define CFE_TIME_SEND_DIAGNOSTIC_TLM_CC 2 /∗ request diagnostic hk telemetry ∗/
- #define CFE_TIME_SET_SOURCE_CC 3 /∗ set clock source (int vs ext) ∗/
- #define CFE_TIME_SET_STATE_CC 4 /∗ set clock state ∗/
- #define CFE_TIME_ADD_DELAY_CC 5 /∗ add tone delay value ∗/
- #define CFE_TIME_SUB_DELAY_CC 6 /∗ sub tone delay value ∗/

- #define CFE_TIME_SET_TIME_CC 7 /* set time */
- #define CFE_TIME_SET_MET_CC 8 /* set MET */
- #define CFE_TIME_SET_STCF_CC 9 /* set STCF */
- #define CFE_TIME_SET_LEAP_SECONDS_CC 10 /* set Leap Seconds */
- #define CFE_TIME_ADD_ADJUST_CC 11 /* add one time STCF adjustment */
- #define CFE_TIME_SUB_ADJUST_CC 12 /* subtract one time STCF adjustment */
- #define CFE_TIME_ADD_1HZ_ADJUSTMENT_CC 13 /* add 1Hz STCF adjustment */
- #define CFE_TIME_SUB_1HZ_ADJUSTMENT_CC 14 /* subtract 1Hz STCF adjustment */
- #define CFE_TIME_SET_SIGNAL_CC 15 /* set clock signal (pri vs red) */

Clock "state flag" values

- #define CFE_TIME_FLAG_CLKSET 0x8000
  The spacecraft time has been set.
- #define CFE_TIME_FLAG_FLYING 0x4000
  This instance of Time Services is flywheeling.
- #define CFE_TIME_FLAG_SRCINT 0x2000
  The clock source is set to "internal".
- #define CFE_TIME_FLAG_SIGPRI 0x1000
  The clock signal is set to "primary".
- #define CFE_TIME_FLAG_SRVFLY 0x0800
  The Time Server is in flywheel mode.
- #define CFE_TIME_FLAG_CMDFLY 0x0400
  This instance of Time Services was commanded into flywheel mode.
- #define CFE_TIME_FLAG_ADDADJ 0x0200
  One time STCF Adjustment is to be done in positive direction.
- #define CFE_TIME_FLAG_ADD1HZ 0x0100
  1 Hz STCF Adjustment is to be done in a positive direction
- #define CFE_TIME_FLAG_ADDTCL 0x0080
  Time Client Latency is applied in a positive direction.
- #define CFE_TIME_FLAG_SERVER 0x0040
  This instance of Time Services is a Time Server.
- #define CFE_TIME_FLAG_GDTONE 0x0020
  The tone received is good compared to the last tone received.
- #define CFE_TIME_FLAG_UNUSED 0x001F
  Reserved flags - should be zero.

Typedefs

- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_Noop_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ResetCounters_t
- typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendDiagnosticTlm_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_AddDelay_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SubDelay_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SetMET_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SetSTCF_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_AddAdjust_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SubAdjust_t
- typedef CFE_TIME_TimeCmd_t CFE_TIME_SetTime_t
- typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Add1HZAdjustment_t
- typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Sub1HZAdjustment_t
- typedef CFE_TIME_HousekeepingTlm_t CFE_TIME_HkPacket_t
- typedef CFE_TIME_DiagnosticTlm_t CFE_TIME_DiagPacket_t

13.72.1 Macro Definition Documentation

13.72.1.1 CFE_TIME_ADD_1HZ_ADJUSTMENT_CC

```
#define CFE_TIME_ADD_1HZ_ADJUSTMENT_CC 13 /* add 1Hz STCF adjustment */
```

Name Add Delta to Spacecraft Time Correlation Factor each 1Hz

Description

This command has been updated to take actual sub-seconds (1/2^32 seconds) rather than micro-seconds as an input argument. This change occurred after the determination was made that one micro-second is too large an increment for a constant 1Hz adjustment.

This command continuously adjusts the Spacecraft Time Correlation Factor (STCF) every second, by adding the specified value. The adjustment to the STCF is applied in the Time Service local 1Hz interrupt handler. As the local 1Hz interrupt is not synchronized to the tone signal, one cannot say when the adjustment will occur, other than once a second, at about the same time relative to the tone.

There was some debate about whether the maximum 1Hz clock drift correction factor would ever need to exceed some small fraction of a second. But, the decision was made to provide the capability to make 1Hz adjustments greater than one second and leave it to the ground system to provide mission specific limits.

Command Mnemonic(s) **`$sc_$cpu_TIME_Add1HzSTCF`**

Command Structure

CFE_TIME_Add1HZAdjustment_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_TIME_STCFSecs`** - Housekeeping Telemetry point indicating new STCF seconds value
- **`$sc_$cpu_TIME_STCFSubsecs`** - Housekeeping Telemetry point indicating new STCF subseconds value
- The CFE_TIME_1HZ_EID informational event message will be generated

Error Conditions

- Platorm receiving the command is not a Time Server

  Evidence of Failure may be found in the following telemetry:
- **`$sc_$cpu_TIME_CMDEC`** - command error counter will increment
- Error specific event message will be issued (CFE_TIME_1HZ_CFG_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_SUB_ADJUST_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC

Definition at line 612 of file cfe_time_msg.h.

13.72.1.2 CFE_TIME_ADD_ADJUST_CC

```
#define CFE_TIME_ADD_ADJUST_CC 11 /* add one time STCF adjustment */
```

Name Add Delta to Spacecraft Time Correlation Factor

Description

This command adjusts the Spacecraft Time Correlation Factor (STCF) by adding the specified value. The new STCF takes effect immediately upon execution of this command.

Command Mnemonic(s) **$sc_$cpu_TIME_AddSTCFAdj**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_STCFSecs -** Housekeeping Telemetry point indicating new STCF seconds value
- **$sc_$cpu_TIME_STCFSubsecs -** Housekeeping Telemetry point indicating new STCF subseconds value
- The CFE_TIME_DELTA_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platorm receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** command error counter will increment
- Error specific event messages will be issued (CFE_TIME_DELTA_ERR_EID or CFE_TIME_DELTA_CFG_↩EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_SUB_ADJUST_CC, CFE_TIME_ADD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUSTMENT_CC

Definition at line 532 of file cfe_time_msg.h.

13.72.1.3 CFE_TIME_ADD_DELAY_CC

```
#define CFE_TIME_ADD_DELAY_CC 5 /* add tone delay value */
```

Name Add Time to Tone Time Delay

Description

This command is used to factor out a known, predictable latency between the Time Server and a particular Time Client. The correction is applied (added) to the current time calculation for Time Clients, so this command has no meaning for Time Servers. Each Time Client can have a unique latency setting. The latency value is a positive number of seconds and microseconds that represent the deviation from the time maintained by the Time Server.

Command Mnemonic(s) **`$sc_$cpu_TIME_AddClockLat`**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_TIME_DLatentS`**, **`$sc_$cpu_TIME_DLatentSs`** - Housekeeping Telemetry point indicating command specified values
- **`$sc_$cpu_TIME_DLatentDir`** - Diagnostic Telemetry point indicating commanded latency direction
- The CFE_TIME_DELAY_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platorm receiving the command is not a Time Client

Evidence of Failure may be found in the following telemetry:

- **`$sc_$cpu_TIME_CMDEC`** - command error counter will increment
- Error specific event messages will be issued (CFE_TIME_DELAY_CFG_EID or CFE_TIME_DELAY_ERR↩_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_SUB_DELAY_CC

Definition at line 302 of file cfe_time_msg.h.

13.72.1.4 CFE_TIME_FLAG_ADD1HZ

```
#define CFE_TIME_FLAG_ADD1HZ 0x0100
```

Definition at line 718 of file cfe_time_msg.h.

13.72.1.5 CFE_TIME_FLAG_ADDADJ

```
#define CFE_TIME_FLAG_ADDADJ 0x0200
```

Definition at line 717 of file cfe_time_msg.h.

13.72.1.6 CFE_TIME_FLAG_ADDTCL

```
#define CFE_TIME_FLAG_ADDTCL 0x0080
```

Definition at line 719 of file cfe_time_msg.h.

13.72.1.7 CFE_TIME_FLAG_CLKSET

```
#define CFE_TIME_FLAG_CLKSET 0x8000
```

Definition at line 711 of file cfe_time_msg.h.

13.72.1.8 CFE_TIME_FLAG_CMDFLY

```
#define CFE_TIME_FLAG_CMDFLY 0x0400
```

Definition at line 716 of file cfe_time_msg.h.

13.72.1.9 CFE_TIME_FLAG_FLYING

```
#define CFE_TIME_FLAG_FLYING 0x4000
```

Definition at line 712 of file cfe_time_msg.h.

13.72.1.10 CFE_TIME_FLAG_GDTONE

```
#define CFE_TIME_FLAG_GDTONE 0x0020
```

Definition at line 721 of file cfe_time_msg.h.

13.72.1.11 CFE_TIME_FLAG_SERVER

```
#define CFE_TIME_FLAG_SERVER 0x0040
```

Definition at line 720 of file cfe_time_msg.h.

13.72.1.12 CFE_TIME_FLAG_SIGPRI

```
#define CFE_TIME_FLAG_SIGPRI 0x1000
```

Definition at line 714 of file cfe_time_msg.h.

13.72.1.13 CFE_TIME_FLAG_SRCINT

```
#define CFE_TIME_FLAG_SRCINT 0x2000
```

Definition at line 713 of file cfe_time_msg.h.

13.72.1.14 CFE_TIME_FLAG_SRVFLY

```
#define CFE_TIME_FLAG_SRVFLY 0x0800
```

Definition at line 715 of file cfe_time_msg.h.

13.72.1.15 CFE_TIME_FLAG_UNUSED

```
#define CFE_TIME_FLAG_UNUSED 0x001F
```

Definition at line 722 of file cfe_time_msg.h.

13.72.1.16 CFE_TIME_NOOP_CC

```
#define CFE_TIME_NOOP_CC 0 /* no-op command */
```

Name Time No-Op

Description

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Time Services task.

Command Mnemonic(s) **`$sc_$cpu_TIME_NOOP`**

Command Structure

CFE_TIME_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- The CFE_TIME_NOOP_EID informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Time Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

None

See also

Definition at line 80 of file cfe_time_msg.h.

13.72.1.17 CFE_TIME_RESET_COUNTERS_CC

```
#define CFE_TIME_RESET_COUNTERS_CC 1 /* reset counters */
```

Name Time Reset Counters

Description

This command resets the following counters within the Time Services Housekeeping Telemetry :

- Command Execution Counter ($sc_$cpu_TIME_CMDPC)
- Command Error Counter ($sc_$cpu_TIME_CMDEC) This command also resets the following counters within the Time Services Diagnostic Telemetry :
- Tone Signal Detected Software Bus Message Counter ($sc_$cpu_TIME_DTSDetCNT)
- Time at the Tone Data Software Bus Message Counter ($sc_$cpu_TIME_DTatTCNT)
- Tone Signal/Data Verify Counter ($sc_$cpu_TIME_DVerifyCNT)
- Tone Signal/Data Error Counter ($sc_$cpu_TIME_DVerifyER)
- Tone Signal Interrupt Counter ($sc_$cpu_TIME_DTsISRCNT)
- Tone Signal Interrupt Error Counter ($sc_$cpu_TIME_DTsISRERR)
- Tone Signal Task Counter ($sc_$cpu_TIME_DTsTaskCNT)
- Local 1 Hz Interrupt Counter ($sc_$cpu_TIME_D1HzISRCNT)
- Local 1 Hz Task Counter ($sc_$cpu_TIME_D1HzTaskCNT)
- Reference Time Version Counter ($sc_$cpu_TIME_DVersionCNT)

Command Mnemonic(s) **`$sc_$cpu_TIME_ResetCtrs`**

Command Structure

CFE_TIME_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- The CFE_TIME_RESET_EID informational event message will be generated

Error Conditions

There are no error conditions for this command. If the Time Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

Criticality

None

See also

Definition at line 124 of file cfe_time_msg.h.

13.72.1.18 CFE_TIME_SEND_DIAGNOSTIC_TLM_CC

```
#define CFE_TIME_SEND_DIAGNOSTIC_TLM_CC 2 /* request diagnostic hk telemetry */
```

Name Request TIME Diagnostic Telemetry

Description

This command requests that the Time Service generate a message containing various data values not included in the normal Time Service housekeeping message. The command requests only a single copy of the diagnostic message. Refer to CFE_TIME_DiagnosticTlm_t for a description of the Time Service diagnostic message contents.

Command Mnemonic(s) **`$sc_$cpu_TIME_RequestDiag`**

Command Structure

CFE_TIME_NoArgsCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- Sequence Counter for CFE_TIME_DiagnosticTlm_t will increment
- The CFE_TIME_DIAG_EID debug event message will be generated

Error Conditions

There are no error conditions for this command. If the Time Services receives the command, the event and telemetry is sent (although one or both may be filtered by EVS and TO) and the counter is incremented unconditionally.

Criticality

None

See also

Definition at line 158 of file cfe_time_msg.h.

13.72.1.19 CFE_TIME_SET_LEAP_SECONDS_CC

```
#define CFE_TIME_SET_LEAP_SECONDS_CC 10 /* set Leap Seconds */
```

Name Set Leap Seconds

Description

This command sets the spacecraft Leap Seconds to the specified value. Leap Seconds may be positive or negative, and there is no limit to the value except, of course, the limit imposed by the 16 bit signed integer data type. The new Leap Seconds value takes effect immediately upon execution of this command.

Command Mnemonic(s) **$sc_$cpu_TIME_SetClockLeap**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_LeapSecs -** Housekeeping Telemetry point indicating new Leap seconds value
- The CFE_TIME_LEAPS_EID informational event message will be generated

Error Conditions

- Platorm receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** command error counter will increment
- Error specific event messages will be issued (CFE_TIME_LEAPS_CFG_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_SET_TIME_CC, CFE_TIME_SET_MET_CC, CFE_TIME_SET_STCF_CC

Definition at line 497 of file cfe_time_msg.h.

13.72.1.20 CFE_TIME_SET_MET_CC

```
#define CFE_TIME_SET_MET_CC 8 /* set MET */
```

Name Set Mission Elapsed Time

Description

This command sets the Mission Elapsed Timer (MET) to the specified value.

Note that the MET (as implemented for cFE Time Service) is a logical representation and not a physical timer. Thus, setting the MET is not dependent on whether the hardware supports a MET register that can be written to.

Note also that Time Service "assumes" that during normal operation, the MET is synchronized to the tone signal. Therefore, unless operating in FLYWHEEL mode, the sub-seconds portion of the MET will be set to zero at the next tone signal interrupt.

The new MET takes effect immediately upon execution of this command.

Command Mnemonic(s) **`$sc_$cpu_TIME_SetClockMET`**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_TIME_METSecs`** - Housekeeping Telemetry point indicating new MET seconds value
- **`$sc_$cpu_TIME_METSubsecs`** - Housekeeping Telemetry point indicating new MET subseconds value
- The CFE_TIME_MET_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platorm receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **`$sc_$cpu_TIME_CMDEC`** - command error counter will increment
- Error specific event messages will be issued (CFE_TIME_MET_CFG_EID or CFE_TIME_MET_ERR_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_SET_TIME_CC, CFE_TIME_SET_STCF_CC, CFE_TIME_SET_LEAP_SECONDS_CC

Definition at line 425 of file cfe_time_msg.h.

13.72.1.21 CFE_TIME_SET_SIGNAL_CC

```
#define CFE_TIME_SET_SIGNAL_CC 15 /* set clock signal (pri vs red) */
```

Name Set Tone Signal Source

Description

This command selects the Time Service tone signal source. Although the list of potential tone signal sources is mission specific, a common choice is the selection of primary or redundant tone signal. The selection may be available to both the Time Server and Time Clients, depending on hardware configuration.

Notes:

- This command is only valid when the CFE_PLATFORM_TIME_CFG_SIGNAL configuration parameter in the cfe_platform_cfg.h file has been set to true.

Command Mnemonic(s) **`$sc_$cpu_TIME_SetSignal`**

Command Structure

CFE_TIME_SetSignal_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_TIME_DSignal`** - Diagnostic Telemetry point will indicate the command specified value
- The CFE_TIME_SIGNAL_EID informational event message will be generated

Error Conditions

- Invalid Signal selection (a value other than CFE_TIME_ToneSignalSelect_PRIMARY or CFE_TIME_ToneSignalSelect_REDUNDANT was specified)
- Multiple Tone Signal Sources not available on this platform

Evidence of failure may be found in the following telemetry:

- **`$sc_$cpu_TIME_CMDEC`** - Command Error counter will increment
- Error specific event message (either CFE_TIME_SIGNAL_CFG_EID or CFE_TIME_SIGNAL_ERR_EID)

Criticality

Although tone signal source selection is important, this command is not critical

See also

CFE_TIME_SET_STATE_CC, CFE_TIME_SET_SOURCE_CC

Definition at line 703 of file cfe_time_msg.h.

13.72.1.22 CFE_TIME_SET_SOURCE_CC

```
#define CFE_TIME_SET_SOURCE_CC 3 /* set clock source (int vs ext) */
```

Name Set Time Source

Description

This command selects the Time Service clock source. Although the list of potential clock sources is mission specific and defined via configuration parameters, this command provides a common method for switching between the local processor clock and an external source for time data.

When commanded to accept external time data (GPS, MET, spacecraft time, etc.), the Time Server will enable input via an API function specific to the configuration definitions for the particular source. When commanded to use internal time data, the Time Server will ignore the external data. However, the Time Server will continue to use the API function as the trigger to generate a "time at the tone" command packet regardless of the internal/external command selection.

Notes:

- Operating in FLYWHEEL mode is not considered a choice related to clock source, but rather an element of the clock state. See below for a description of the CFE_TIME_SET_STATE_CC command.
- This command is only valid when the CFE_PLATFORM_TIME_CFG_SOURCE configuration parameter in the cfe_platform_cfg.h file has been set to true.

Command Mnemonic(s) **$sc_$cpu_TIME_SetSource**

Command Structure

CFE_TIME_SetSource_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_DSource -** Diagnostic Telemetry point will indicate the command specified value
- The CFE_TIME_SOURCE_EID informational event message will be generated

Error Conditions

- Invalid Source selection (a value other than CFE_TIME_SourceSelect_INTERNAL or CFE_TIME_Source↩ Select_EXTERNAL was specified)
- Time source selection not allowed on this platform

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** Command Error counter will increment
- Error specific event message (either CFE_TIME_SOURCE_CFG_EID or CFE_TIME_SOURCE_ERR_EID)

Criticality

Although clock source selection is important, this command is not critical.

See also

CFE_TIME_SET_STATE_CC, CFE_TIME_SET_SIGNAL_CC

Definition at line 208 of file cfe_time_msg.h.

13.72.1.23 CFE_TIME_SET_STATE_CC

```
#define CFE_TIME_SET_STATE_CC 4 /* set clock state */
```

Name Set Time State

Description

This command indirectly affects the Time Service on-board determination of clock state. Clock state is a combination of factors, most significantly whether the spacecraft time has been accurately set, and whether Time Service is operating in FLYWHEEL mode.

This command may be used to notify the Time Server that spacecraft time is now correct, or that time is no longer correct. This information will be distributed to Time Clients, and in turn, to any interested sub-systems.

Also, this command may be used to force a Time Server or Time Client into FLYWHEEL mode. Use of FLYWHEEL mode is mainly for debug purposes although in extreme circumstances, it may be of value to force Time Service not to rely on normal time updates. Note that when commanded into FLYWHEEL mode, the Time Service will remain so until receipt of another "set state" command setting the state into a mode other than FLYWHEEL.

Note also that setting the clock state to VALID or INVALID on a Time Client that is currently getting time updates from the Time Server will have very limited effect. As soon as the Time Client receives the next time update, the VALID/INVALID selection will be set to that of the Time Server. However, setting a Time Client to FLYWHEEL cannot be overridden by the Time Server since the Time Client will ignore time updates from the Time Server while in FLYWHEEL mode.

Command Mnemonic(s) **$sc_$cpu_TIME_SetState**

Command Structure

CFE_TIME_SetState_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_StateFlg, $sc_$cpu_TIME_FlagSet, $sc_$cpu_TIME_FlagFly, $sc_$cpu_TIME_FlagSrc, $sc_$cpu_TIME_FlagPri, $sc_$cpu_TIME_FlagSfly, $sc_$cpu_TIME_FlagCfly, $sc_$cpu_TIME_FlagAdjd, $sc_$cpu_TIME_Flag1Hzd, $sc_$cpu_TIME_FlagClat, $sc_$cpu_TIME_FlagSorC, $sc_$cpu_TIME_FlagNIU -** Housekeeping Telemetry point "may"l indicate the command specified value (see above)
- The CFE_TIME_STATE_EID informational event message will be generated

Error Conditions

- Invalid State selection (a value other than CFE_TIME_ClockState_INVALID, CFE_TIME_ClockState_VALID or CFE_TIME_ClockState_FLYWHEEL was specified)
- Time source selection not allowed on this platform

Evidence of failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** Command Error counter will increment

- Error specific event message (CFE_TIME_STATE_ERR_EID)

Criticality

Setting Time Service into FLYWHEEL mode is not particularly hazardous, as the result may be that the calculation of spacecraft time is done using a less than optimal timer. However, inappropriately setting the clock state to V↩ ALID (indicating that spacecraft time is accurate) may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_SET_SOURCE_CC, CFE_TIME_SET_SIGNAL_CC

Definition at line 264 of file cfe_time_msg.h.

13.72.1.24 CFE_TIME_SET_STCF_CC

```
#define CFE_TIME_SET_STCF_CC 9 /* set STCF */
```

Name Set Spacecraft Time Correlation Factor

Description

This command sets the Spacecraft Time Correlation Factor (STCF) to the specified value. This command differs from the previously described SET CLOCK in the nature of the command argument. This command sets the STCF value directly, rather than extracting the STCF from a value representing the total of MET, STCF and optionally, Leap Seconds. The new STCF takes effect immediately upon execution of this command.

Command Mnemonic(s) **`$sc_$cpu_TIME_SetClockSTCF`**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC`** - command execution counter will increment
- **`$sc_$cpu_TIME_STCFSecs`** - Housekeeping Telemetry point indicating new STCF seconds value
- **`$sc_$cpu_TIME_STCFSubsecs`** - Housekeeping Telemetry point indicating new STCF subseconds value
- The CFE_TIME_STCF_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)

- Platorm receiving the command is not a Time Server

  Evidence of Failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** command error counter will increment
- Error specific event messages will be issued (CFE_TIME_STCF_CFG_EID or CFE_TIME_STCF_ERR_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_SET_TIME_CC, CFE_TIME_SET_MET_CC, CFE_TIME_SET_LEAP_SECONDS_CC

Definition at line 462 of file cfe_time_msg.h.

13.72.1.25 CFE_TIME_SET_TIME_CC

```
#define CFE_TIME_SET_TIME_CC 7 /* set time */
```

Name Set Spacecraft Time

Description

This command sets the spacecraft clock to a new value, regardless of the current setting (time jam). The new time value represents the desired offset from the mission-defined time epoch and takes effect immediately upon execution of this command. Time Service will calculate a new STCF value based on the current MET and the desired new time using one of the following:

If Time Service is configured to compute current time as TAI

- **STCF = (new time) - (current MET)**
- **(current time) = (current MET) + STCF**

  If Time Service is configured to compute current time as UTC

- **STCF = ((new time) - (current MET)) + (Leap Seconds)**
- **(current time) = ((curent MET) + STCF) - (Leap Seconds)**

Command Mnemonic(s) **$sc_$cpu_TIME_SetClock**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_STCFSecs -** Housekeeping Telemetry point indicating newly calculated STCF seconds value
- **$sc_$cpu_TIME_STCFSubsecs -** Housekeeping Telemetry point indicating newly calculated STCF sub-seconds value
- The CFE_TIME_TIME_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platorm receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** command error counter will increment
- Error specific event messages will be issued (CFE_TIME_TIME_CFG_EID or CFE_TIME_TIME_ERR_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_SET_MET_CC, CFE_TIME_SET_STCF_CC, CFE_TIME_SET_LEAP_SECONDS_CC

Definition at line 385 of file cfe_time_msg.h.

13.72.1.26 CFE_TIME_SUB_1HZ_ADJUSTMENT_CC

```
#define CFE_TIME_SUB_1HZ_ADJUSTMENT_CC 14 /* subtract 1Hz STCF adjustment */
```

Name Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

Description

This command has been updated to take actual sub-seconds (1/2^32 seconds) rather than micro-seconds as an input argument. This change occurred after the determination was made that one micro-second is too large an increment for a constant 1Hz adjustment.

This command continuously adjusts the Spacecraft Time Correlation Factor (STCF) every second, by subtracting the specified value. The adjustment to the STCF is applied in the Time Service local 1Hz interrupt handler. As the local 1Hz interrupt is not synchronized to the tone signal, one cannot say when the adjustment will occur, other than once a second, at about the same time relative to the tone.

There was some debate about whether the maximum 1Hz clock drift correction factor would ever need to exceed some small fraction of a second. But, the decision was made to provide the capability to make 1Hz adjustments greater than one second and leave it to the ground system to provide mission specific limits.

Command Mnemonic(s) **$sc_$cpu_TIME_Sub1HzSTCF**

Command Structure

CFE_TIME_Sub1HZAdjustment_t

Command Verification

Successful execution of this command may be verified with the following telemetry: Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_STCFSecs -** Housekeeping Telemetry point indicating new STCF seconds value
- **$sc_$cpu_TIME_STCFSubsecs -** Housekeeping Telemetry point indicating new STCF subseconds value
- The CFE_TIME_1HZ_EID informational event message will be generated

Error Conditions

- Platorm receiving the command is not a Time Server

  Evidence of Failure may be found in the following telemetry:
- **$sc_$cpu_TIME_CMDEC -** command error counter will increment
- Error specific event message will be issued (CFE_TIME_1HZ_CFG_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_SUB_ADJUST_CC, CFE_TIME_ADD_1HZ_ADJUSTMENT_CC

Definition at line 660 of file cfe_time_msg.h.

13.72.1.27 CFE_TIME_SUB_ADJUST_CC

```
#define CFE_TIME_SUB_ADJUST_CC 12 /* subtract one time STCF adjustment */
```

Name Subtract Delta from Spacecraft Time Correlation Factor

Description

This command adjusts the Spacecraft Time Correlation Factor (STCF) by subtracting the specified value. The new STCF takes effect immediately upon execution of this command.

Command Mnemonic(s) **$sc_$cpu_TIME_SubSTCFAdj**

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **$sc_$cpu_TIME_CMDPC -** command execution counter will increment
- **$sc_$cpu_TIME_STCFSecs -** Housekeeping Telemetry point indicating new STCF seconds value
- **$sc_$cpu_TIME_STCFSubsecs -** Housekeeping Telemetry point indicating new STCF subseconds value
- The CFE_TIME_DELTA_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platorm receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **$sc_$cpu_TIME_CMDEC -** command error counter will increment
- Error specific event messages will be issued (CFE_TIME_DELTA_ERR_EID or CFE_TIME_DELTA_CFG_↩EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_ADD_ADJUST_CC, CFE_TIME_ADD_1HZ_ADJUSTMENT_CC, CFE_TIME_SUB_1HZ_ADJUST↩MENT_CC

Definition at line 566 of file cfe_time_msg.h.

13.72.1.28 CFE_TIME_SUB_DELAY_CC

```
#define CFE_TIME_SUB_DELAY_CC 6 /* sub tone delay value */
```

Name Subtract Time from Tone Time Delay

Description

This command is used to factor out a known, predictable latency between the Time Server and a particular Time Client. The correction is applied (subtracted) to the current time calculation for Time Clients, so this command has no meaning for Time Servers. Each Time Client can have a unique latency setting. The latency value is a positive number of seconds and microseconds that represent the deviation from the time maintained by the Time Server.

Note that it is unimaginable that the seconds value will ever be anything but zero.

Command Mnemonic(s) `$sc_$cpu_TIME_SubClockLat`

Command Structure

CFE_TIME_TimeCmd_t

Command Verification

Successful execution of this command may be verified with the following telemetry:

- **`$sc_$cpu_TIME_CMDPC` -** command execution counter will increment
- **`$sc_$cpu_TIME_DLatentS`, `$sc_$cpu_TIME_DLatentSs` -** Housekeeping Telemetry point indicating command specified values
- **`$sc_$cpu_TIME_DLatentDir` -** Diagnostic Telemetry point indicating commanded latency direction
- The CFE_TIME_DELAY_EID informational event message will be generated

Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platorm receiving the command is not a Time Client

Evidence of Failure may be found in the following telemetry:

- **`$sc_$cpu_TIME_CMDEC` -** command error counter will increment
- Error specific event messages will be issued (CFE_TIME_DELAY_CFG_EID or CFE_TIME_DELAY_ERR_EID)

Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

See also

CFE_TIME_ADD_DELAY_CC

Definition at line 340 of file cfe_time_msg.h.

13.72.2 Typedef Documentation

13.72.2.1 CFE_TIME_Add1HZAdjustment_t

```
typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Add1HZAdjustment_t
```

Definition at line 865 of file cfe_time_msg.h.

13.72.2.2 CFE_TIME_AddAdjust_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_AddAdjust_t
```

Definition at line 838 of file cfe_time_msg.h.

13.72.2.3 CFE_TIME_AddDelay_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_AddDelay_t
```

Definition at line 834 of file cfe_time_msg.h.

13.72.2.4 CFE_TIME_DiagPacket_t

```
typedef CFE_TIME_DiagnosticTlm_t CFE_TIME_DiagPacket_t
```

Definition at line 1157 of file cfe_time_msg.h.

13.72.2.5 CFE_TIME_HkPacket_t

```
typedef CFE_TIME_HousekeepingTlm_t CFE_TIME_HkPacket_t
```

Definition at line 1156 of file cfe_time_msg.h.

13.72.2.6 CFE_TIME_Noop_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_Noop_t
```

Definition at line 742 of file cfe_time_msg.h.

13.72.2.7 CFE_TIME_ResetCounters_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ResetCounters_t
```

Definition at line 743 of file cfe_time_msg.h.

13.72.2.8 CFE_TIME_SendDiagnosticTlm_t

```
typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendDiagnosticTlm_t
```

Definition at line 744 of file cfe_time_msg.h.

13.72.2.9 CFE_TIME_SetMET_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetMET_t
```

Definition at line 836 of file cfe_time_msg.h.

13.72.2.10 CFE_TIME_SetSTCF_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetSTCF_t
```

Definition at line 837 of file cfe_time_msg.h.

13.72.2.11 CFE_TIME_SetTime_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SetTime_t
```

Definition at line 840 of file cfe_time_msg.h.

13.72.2.12 CFE_TIME_Sub1HZAdjustment_t

```
typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Sub1HZAdjustment_t
```

Definition at line 866 of file cfe_time_msg.h.

13.72.2.13 CFE_TIME_SubAdjust_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SubAdjust_t
```

Definition at line 839 of file cfe_time_msg.h.

13.72.2.14 CFE_TIME_SubDelay_t

```
typedef CFE_TIME_TimeCmd_t CFE_TIME_SubDelay_t
```

Definition at line 835 of file cfe_time_msg.h.

13.73 cfe/fsw/cfe-core/src/inc/cfe_version.h File Reference

```
#include <target_config.h>
```

Macros

- #define CFE_MAJOR_VERSION 6
- #define CFE_MINOR_VERSION 7
- #define CFE_REVISION 8

13.73.1 Macro Definition Documentation

13.73.1.1 CFE_MAJOR_VERSION

```
#define CFE_MAJOR_VERSION 6
```

Definition at line 75 of file cfe_version.h.

Referenced by CFE_ES_NoopCmd(), CFE_ES_TaskInit(), CFE_EVS_NoopCmd(), CFE_EVS_TaskInit(), and CFE_↩ SB_NoopCmd().

13.73.1.2 CFE_MINOR_VERSION

```
#define CFE_MINOR_VERSION 7
```

Definition at line 76 of file cfe_version.h.

Referenced by CFE_ES_NoopCmd(), CFE_ES_TaskInit(), CFE_EVS_NoopCmd(), CFE_EVS_TaskInit(), and CFE_↩ SB_NoopCmd().

13.73.1.3 CFE_REVISION

```
#define CFE_REVISION 8
```

Definition at line 77 of file cfe_version.h.

Referenced by CFE_ES_NoopCmd(), CFE_ES_TaskInit(), CFE_EVS_NoopCmd(), CFE_EVS_TaskInit(), and CFE_↩ SB_NoopCmd().

13.74 cfe/fsw/cfe-core/src/inc/network_includes.h File Reference

13.75 cfe/fsw/cfe-core/src/inc/private/cfe_es_erlog_typedef.h File Reference

```
#include <common_types.h>
#include <cfe_time.h>
```

Data Structures

- struct CFE_ES_DebugVariables_t
- struct CFE_ES_ERLog_t

13.75.1 Detailed Description

Created on: Jan 22, 2015 Author: joseph.p.hickey@nasa.gov

Definition of the CFE_ES_ERLog structure type. This was moved into its own header file since it is referenced by multiple CFE core apps.

13.76 cfe/fsw/cfe-core/src/inc/private/cfe_es_perfdata_typedef.h File Reference

```
#include <common_types.h>
#include "cfe_mission_cfg.h"
#include "cfe_platform_cfg.h"
```

Data Structures

- struct CFE_ES_PerfDataEntry_t
- struct CFE_ES_PerfMetaData_t
- struct CFE_ES_PerfData_t

Macros

- #define CFE_ES_PERF_32BIT_WORDS_IN_MASK ((CFE_MISSION_ES_PERF_MAX_IDS) / 32)

13.76.1 Detailed Description

Created on: Jan 22, 2015 Author: joseph.p.hickey@nasa.gov

Placeholder for file content description

13.76.2 Macro Definition Documentation

13.76.2.1 CFE_ES_PERF_32BIT_WORDS_IN_MASK

```
#define CFE_ES_PERF_32BIT_WORDS_IN_MASK ((CFE_MISSION_ES_PERF_MAX_IDS) / 32)
```

Definition at line 37 of file cfe_es_perfdata_typedef.h.

Referenced by CFE_ES_SetPerfFilterMaskCmd(), CFE_ES_SetPerfTriggerMaskCmd(), and CFE_ES_SetupPerf↩ Variables().

13.77 cfe/fsw/cfe-core/src/inc/private/cfe_es_resetdata_typedef.h File Reference

```
#include <common_types.h>
#include <cfe_time.h>
#include "cfe_es_erlog_typedef.h"
#include "cfe_es_perfdata_typedef.h"
#include "cfe_evs_log_typedef.h"
#include "cfe_platform_cfg.h"
```

Data Structures

- struct CFE_ES_ResetVariables_t
- struct CFE_ES_ResetData_t

13.77.1 Detailed Description

Created on: Jan 22, 2015 Author: joseph.p.hickey@nasa.gov

Definition of the CFE_ES_ResetData structure type. This was moved into its own header file since it is referenced by multiple CFE core apps.

13.78 cfe/fsw/cfe-core/src/inc/private/cfe_evs_log_typedef.h File Reference

```
#include <common_types.h>
#include "cfe_evs_msg.h"
```

Data Structures

- struct CFE_EVS_Log_t

13.78.1 Detailed Description

Created on: Jan 22, 2015 Author: joseph.p.hickey@nasa.gov

Definition of the CFE_EVS_Log structure type. This was moved into its own header file since it is referenced by multiple CFE core apps.

13.79 cfe/fsw/cfe-core/src/inc/private/cfe_private.h File Reference

```
#include "common_types.h"
#include "cfe.h"
#include "cfe_platform_cfg.h"
```

Functions

- void CFE_TIME_TaskMain (void)

  Entry Point for cFE Core Application.
- void CFE_SB_TaskMain (void)

  Entry Point for cFE Core Application.
- void CFE_EVS_TaskMain (void)

  Entry Point for cFE Core Application.
- void CFE_ES_TaskMain (void)

  Entry Point for cFE Core Application.
- void CFE_TBL_TaskMain (void)

  Entry Point for cFE Table Services Core Application.
- int32 CFE_EVS_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_SB_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_TIME_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_TBL_EarlyInit (void)

  Initializes the Table Services API Library.
- int32 CFE_ES_CDS_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_FS_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_TBL_CleanUpApp (uint32 AppId)

  Removes TBL resources associated with specified Application.
- int32 CFE_SB_CleanUpApp (uint32 AppId)

Removes SB resources associated with specified Application.

- int32 CFE_EVS_CleanUpApp (uint32 AppId)

  Removes EVS resources associated with specified Application.

- int32 CFE_TIME_CleanUpApp (uint32 AppId)

  Removes TIME resources associated with specified Application.

- int32 CFE_ES_RegisterCDSEx (CFE_ES_CDSHandle_t *HandlePtr, int32 BlockSize, const char *Name, bool CriticalTbl)

  Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)

- int32 CFE_ES_DeleteCDS (const char *CDSName, bool CalledByTblServices)

  Deletes the specified CDS from the CDS Registry and frees CDS Memory.

13.79.1 Function Documentation

13.79.1.1 CFE_ES_CDS_EarlyInit()

```
int32 CFE_ES_CDS_EarlyInit (
            void  )
```

Description

Initializes the cFE core module API Library

Assumptions, External Events, and Notes:

1. This function MUST be called before any module API's are called.

Initializes the cFE core module API Library.

Description

Locates and validates any pre-existing CDS memory or initializes the memory as a fresh CDS.

Assumptions, External Events, and Notes:

None

SysLog Messages

Returns

None

Definition at line 149 of file cfe_es_cds.c.

13.79.1.2 CFE_ES_DeleteCDS()

```
int32 CFE_ES_DeleteCDS (
            const char * CDSName,
            bool CalledByTblServices )
```

Description

Removes the record of the specified CDS from the CDS Registry and frees the associated CDS memory for future use.

Assumptions, External Events, and Notes:

None

Parameters

in	*CDSName*	- Pointer to character string containing complete CDS Name (of the format "AppName.CDSName").
in	*CalledByTblServices*	- Flag that identifies whether the CDS is supposed to be a Critical Table Image or not.

Returns

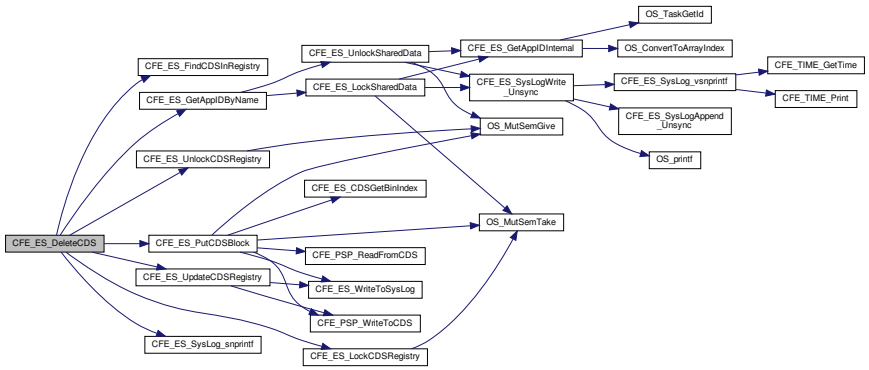
CFE_SUCCESS Operation was performed successfully
CFE_ES_CDS_WRONG_TYPE_ERR Occurs when Table Services is trying to delete a Critical Data Store that is not a Critical Table Image or when Executive Services is trying to delete a Critical Table Image.
CFE_ES_CDS_OWNER_ACTIVE_ERR Occurs when an attempt was made to delete a CDS when an application with the same name associated with the CDS is still present. CDSs can ONLY be deleted when Applications that created them are not present in the system.
CFE_ES_CDS_NOT_FOUND_ERR Occurs when a search of the Critical Data Store Registry does not find a critical data store with the specified name.
Any of the return values from CFE_ES_UpdateCDSRegistry
Any of the return values from CFE_ES_PutCDSBlock

Definition at line 752 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_NOT_FOUND, CFE_ES_CDS_NOT_FOUND_ERR, CFE_ES_CDS_OWNER_ACTIVE_ERR, CFE_ES_CDS_WRONG_TYPE_ERR, CFE_ES_ERR_APPNAME, CFE_ES_FindCDSInRegistry(), CFE_ES_GetAppIDByName(), CFE_ES_Global, CFE_ES_LockCDSRegistry(), CFE_ES_MAX_SYSLOG_MSG_SIZE, CFE_ES_PutCDSBlock(), CFE_ES_SYSLOG_APPEND, CFE_ES_SysLog_snprintf(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UpdateCDSRegistry(), CFE_SUCCESS, CFE_ES_CDS_RegRec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, NULL, OS_MAX_API_NAME, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_DeleteCDSCmd().

Here is the call graph for this function:



13.79.1.3 CFE_ES_RegisterCDSEx()

```
int32 CFE_ES_RegisterCDSEx (
            CFE_ES_CDSHandle_t * HandlePtr,
            int32 BlockSize,
            const char * Name,
            bool CriticalTbl )
```

cFE Core task other function call prototypes

Description

This routine is identical to CFE_ES_RegisterCDS except it identifies the contents of the CDS as a critical table. This is crucial because a critical table CDS must only be deleted by cFE Table Services, not via an ES delete CDS command. Otherwise, Table Services may be out of sync with the contents of the CDS.

Assumptions, External Events, and Notes:

1. This function assumes input parameters are error free and have met size/value restrictions.
2. The calling function is responsible for issuing any event messages associated with errors.

Parameters

in	*HandlePtr*	Pointer Application's variable that will contain the CDS Memory Block Handle.
in	*BlockSize*	The number of bytes needed in the CDS.
in	*Name*	Pointer to character string containing the Application's local name for the CDS.
in	*CriticalTbl*	Indicates whether the CDS is to be used as a Critical Table or not
out	∗*HandlePtr*	The handle of the CDS block that can be used in CFE_ES_CopyToCDS and CFE_ES_RestoreFromCDS.

Returns

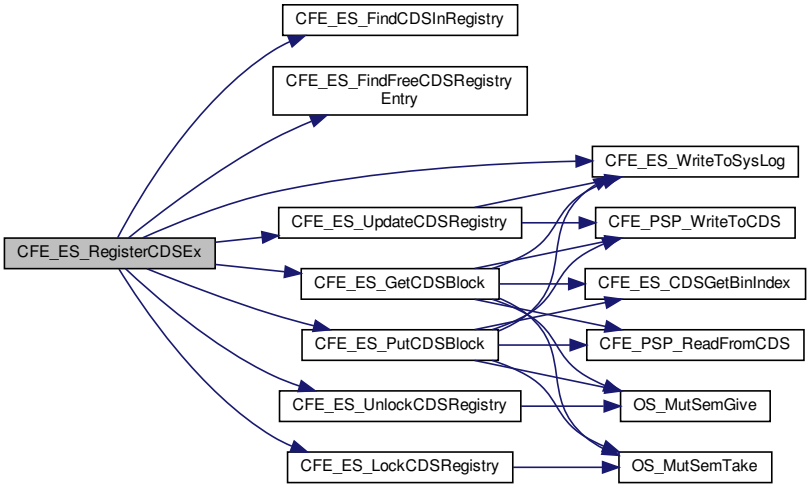
See return codes for CFE_ES_RegisterCDS

Definition at line 230 of file cfe_es_cds.c.

References CFE_ES_Global_t::CDSVars, CFE_ES_CDS_ALREADY_EXISTS, CFE_ES_CDS_MAX_FULL_NAME_LEN, CFE_ES_CDS_NOT_FOUND, CFE_ES_CDS_REGISTRY_FULL, CFE_ES_FindCDSInRegistry(), CFE_ES_FindFreeCDSRegistryEntry(), CFE_ES_GetCDSBlock(), CFE_ES_Global, CFE_ES_LockCDSRegistry(), CFE_ES_PutCDSBlock(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UpdateCDSRegistry(), CFE_ES_WriteToSysLog(), CFE_SUCCESS, CFE_ES_CDS_RegRec_t::MemHandle, CFE_ES_CDS_RegRec_t::Name, NULL, CFE_ES_CDSVariables_t::Registry, CFE_ES_CDS_RegRec_t::Size, CFE_ES_CDS_RegRec_t::Table, and CFE_ES_CDS_RegRec_t::Taken.

Referenced by CFE_ES_RegisterCDS().

Here is the call graph for this function:



13.79.1.4 CFE_ES_TaskMain()

```
void CFE_ES_TaskMain (
            void  )
```

Description

This is the entry point to the cFE ES Core Application.

Assumptions, External Events, and Notes:

None

Return values

None	

Definition at line 80 of file cfe_es_task.c.

13.79.1.5 CFE_EVS_CleanUpApp()

```
int32 CFE_EVS_CleanUpApp (
            uint32 AppId )
```

Description

This function is called by cFE Executive Services to cleanup after an Application has been terminated. It frees resources that have been allocated to the specified Application.

Definition at line 184 of file cfe_evs_task.c.

References CFE_EVS_GlobalData_t::AppData, CFE_EVS_APP_ILLEGAL_APP_ID, CFE_PLATFORM_ES_MAX_A↩PPLICATIONS, CFE_SUCCESS, and EVS_AppData_t::RegisterFlag.

Referenced by CFE_ES_CleanUpApp().

13.79.1.6 CFE_EVS_EarlyInit()

```
int32 CFE_EVS_EarlyInit (
            void  )
```

cFE Core task early init prototypes

Description

Initializes the cFE core module API Library

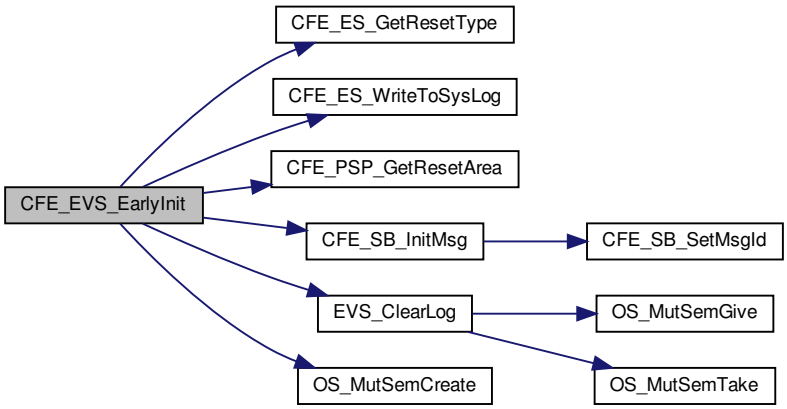
Assumptions, External Events, and Notes:

1. This function MUST be called before any module API's are called.

Definition at line 74 of file cfe_evs_task.c.

References CFE_ES_GetResetType(), CFE_ES_WriteToSysLog(), CFE_EVS_HK_TLM_MID, CFE_EVS_LogMode_DISCARD, CFE_EVS_LogMode_OVERWRITE, CFE_EVS_RESET_AREA_POINTER, CFE_EVS_UNDEF_APPID, CFE_PLATFORM_EVS_DEFAULT_LOG_MODE, CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE, CFE_PLATFORM_EVS_LOG_MAX, CFE_PLATFORM_EVS_PORT_DEFAULT, CFE_PSP_GetResetArea(), CFE_PSP_RST_TYPE_POWERON, CFE_PSP_SUCCESS, CFE_SB_InitMsg(), CFE_SUCCESS, CFE_EVS_GlobalData_t::EVS_AppID, EVS_ClearLog(), CFE_EVS_GlobalData_t::EVS_LogPtr, CFE_EVS_GlobalData_t::EVS_SharedDataMutexID, CFE_EVS_GlobalData_t::EVS_TlmPkt, CFE_EVS_Log_t::LogCount, CFE_EVS_HousekeepingTlm_Payload_t::LogEnabled, CFE_EVS_Log_t::LogFullFlag, CFE_EVS_HousekeepingTlm_Payload_t::LogFullFlag, CFE_EVS_Log_t::LogMode, CFE_EVS_HousekeepingTlm_Payload_t::LogMode, CFE_EVS_Log_t::LogOverflowCounter, CFE_EVS_HousekeepingTlm_Payload_t::MessageFormatMode, CFE_EVS_Log_t::Next, NULL, OS_MutSemCreate(), OS_SUCCESS, CFE_EVS_HousekeepingTlm_Payload_t::OutputPort, and CFE_EVS_HousekeepingTlm_t::Payload.

Here is the call graph for this function:



13.79.1.7 CFE_EVS_TaskMain()

```
void CFE_EVS_TaskMain (
             void  )
```

Description

This is the entry point to the cFE EVS Core Application.

Assumptions, External Events, and Notes:

None

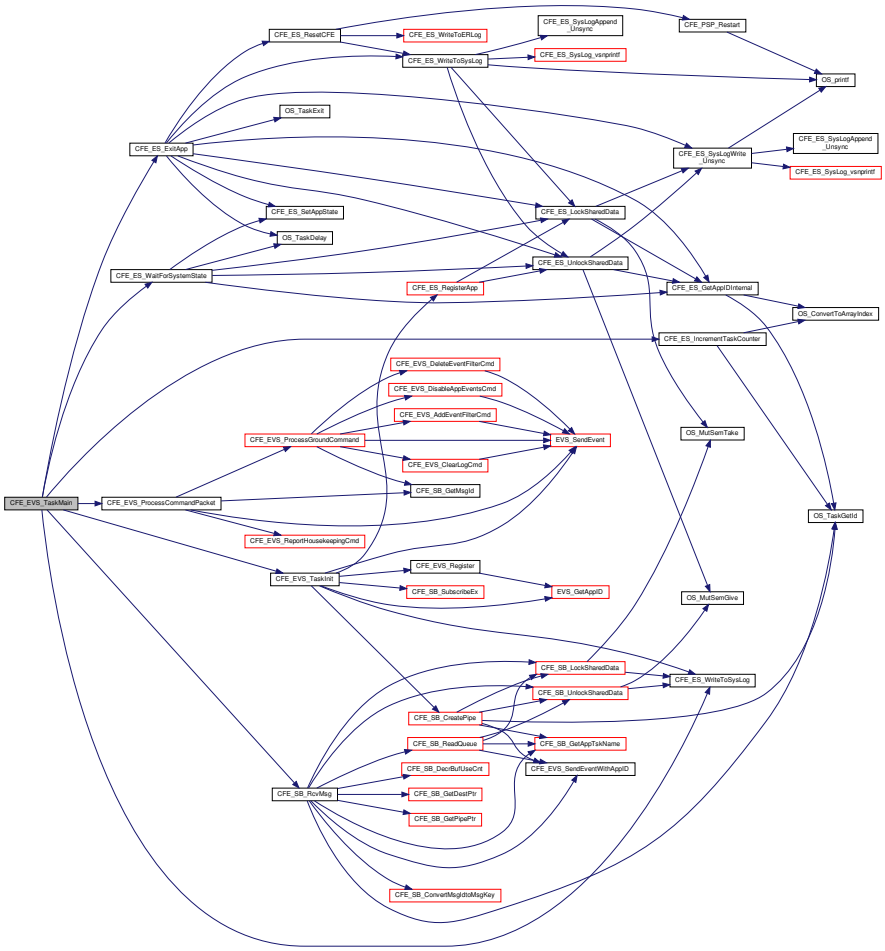
Return values

None	

Definition at line 212 of file cfe_evs_task.c.

References CFE_ES_ExitApp(), CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, CFE_ES_SystemState_CORE_READY, CFE_ES_WaitForSystemState(), CFE_ES_WriteToSysLog(), CFE_EVS_ProcessCommandPacket(), CFE_EVS_TaskInit(), CFE_MISSION_EVS_MAIN_PERF_ID, CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_SB_PEND_FOREVER, CFE_SB_RcvMsg(), CFE_SUCCESS, and CFE_EVS_GlobalData_t::EVS_CommandPipe.

Here is the call graph for this function:



13.79.1.8 CFE_FS_EarlyInit()

```
int32 CFE_FS_EarlyInit (
            void  )
```

Description

Initializes the cFE core module API Library

Assumptions, External Events, and Notes:

1. This function MUST be called before any module API's are called.

13.79.1.9 CFE_SB_CleanUpApp()

```
int32 CFE_SB_CleanUpApp (
            uint32 AppId )
```

Description

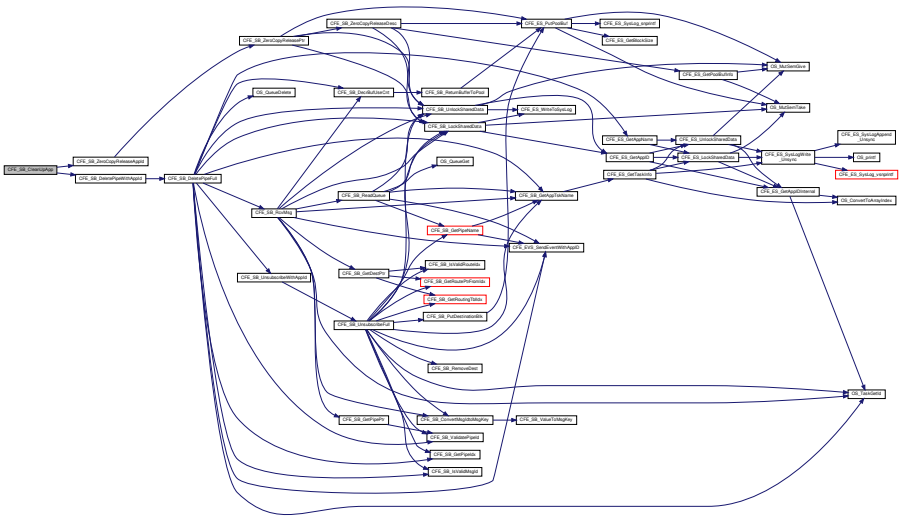
This function is called by cFE Executive Services to cleanup after an Application has been terminated. It frees resources that have been allocated to the specified Application.

Definition at line 126 of file cfe_sb_priv.c.

References CFE_SB_PipeD_t::AppId, CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_DeletePipeWithAppId(), CFE_SB_IN_USE, CFE_SB_ZeroCopyReleaseAppId(), CFE_SUCCESS, CFE_SB_PipeD_t::InUse, CFE_SB_PipeD_t::PipeId, and cfe_sb_t::PipeTbl.

Referenced by CFE_ES_CleanUpApp().

Here is the call graph for this function:



13.79.1.10 CFE_SB_EarlyInit()

```
int32 CFE_SB_EarlyInit (
            void  )
```

Description

Initializes the cFE core module API Library

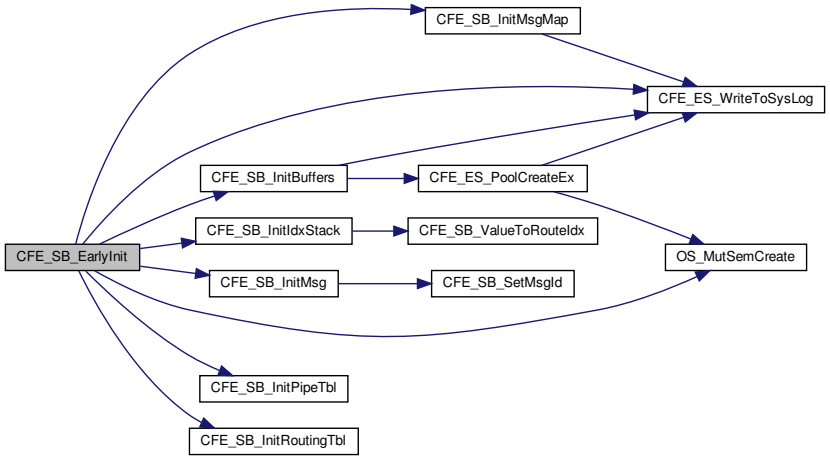
Assumptions, External Events, and Notes:

1. This function MUST be called before any module API's are called.

Definition at line 86 of file cfe_sb_init.c.

References CFE_ES_WriteToSysLog(), CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER, CFE_SB, CFE_SB↩_Default_Qos, CFE_SB_DISABLE, CFE_SB_InitBuffers(), CFE_SB_InitIdxStack(), CFE_SB_InitMsg(), CFE_SB_Init↩MsgMap(), CFE_SB_InitPipeTbl(), CFE_SB_InitRoutingTbl(), CFE_SB_QOS_LOW_PRIORITY, CFE_SB_QOS_LO↩W_RELIABILITY, CFE_SB_STATS_TLM_MID, CFE_SUCCESS, NULL, OS_MutSemCreate(), OS_SUCCESS, CFE↩_SB_Qos_t::Priority, CFE_SB_Qos_t::Reliability, cfe_sb_t::SenderReporting, cfe_sb_t::SharedDataMutexId, cfe_sb_↩t::StatTlmMsg, cfe_sb_t::SubscriptionReporting, and cfe_sb_t::ZeroCopyTail.

Here is the call graph for this function:



13.79.1.11 CFE_SB_TaskMain()

```
void CFE_SB_TaskMain (
            void  )
```

Description

This is the entry point to the cFE SB Core Application.

Assumptions, External Events, and Notes:

None

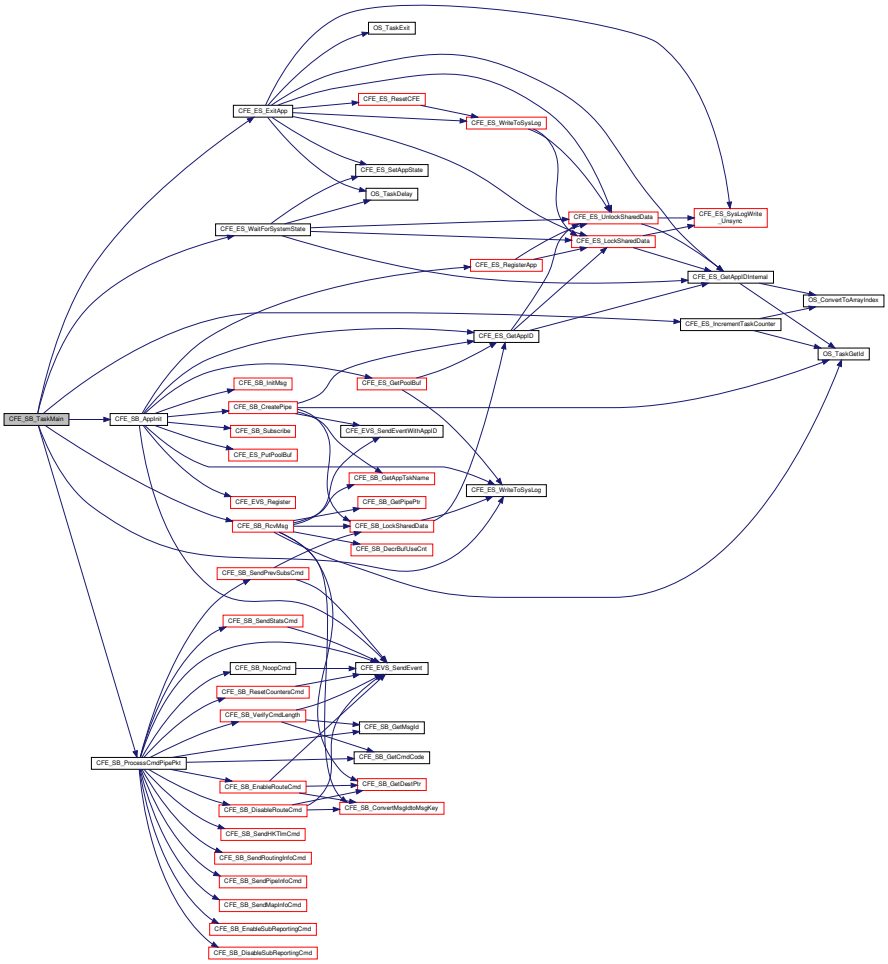
Return values

None	

Definition at line 65 of file cfe_sb_task.c.

References CFE_ES_ExitApp(), CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, CFE_ES_SystemState_CORE_READY, CFE_ES_WaitForSystemState(), CFE_ES_WriteToSysLog(), CFE_MISSION_SB_MAIN_PERF_ID, CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_SB_AppInit(), CFE_SB_PEND_FOREVER, CFE_SB_ProcessCmdPipePkt(), CFE_SB_RcvMsg(), CFE_SUCCESS, cfe_sb_t::CmdPipe, and cfe_sb_t::CmdPipePktPtr.

Here is the call graph for this function:



13.79.1.12 CFE_TBL_CleanUpApp()

```
int32 CFE_TBL_CleanUpApp (
            uint32 AppId )
```

cFE Core task clean up prototypes

Description

This function is called by cFE Executive Services to cleanup after an Application has been terminated. It frees TBL services resources that have been allocated to the specified Application.

Assumptions, External Events, and Notes:

1. This function DOES NOT remove any critical tables associated with the specified application from the Critical Data Store.

Referenced by CFE_ES_CleanUpApp().

13.79.1.13 CFE_TBL_EarlyInit()

```
int32 CFE_TBL_EarlyInit (
            void  )
```

Description

Initializes the Table Services API Library

Assumptions, External Events, and Notes:

1. This function MUST be called before any TBL API's are called.

13.79.1.14 CFE_TBL_TaskMain()

```
void CFE_TBL_TaskMain (
            void  )
```

Description

This is the entry point to the cFE Table Services Core Application. This Application provides the ground interface to the cFE Table Services.

Assumptions, External Events, and Notes:

None

Return values

None	

13.79.1.15 CFE_TIME_CleanUpApp()

```
int32 CFE_TIME_CleanUpApp (
            uint32 AppId )
```

Description

This function is called by cFE Executive Services to cleanup after an Application has been terminated. It frees resources that have been allocated to the specified Application.

Referenced by CFE_ES_CleanUpApp().

13.79.1.16 CFE_TIME_EarlyInit()

```
int32 CFE_TIME_EarlyInit (
            void  )
```

Description

Initializes the cFE core module API Library

Assumptions, External Events, and Notes:

1. This function MUST be called before any module API's are called.

13.79.1.17 CFE_TIME_TaskMain()

```
void CFE_TIME_TaskMain (
            void  )
```

cFE Core task entry point prototypes

Description

This is the entry point to the cFE TIME Core Application.

Assumptions, External Events, and Notes:

None

Return values

None	

13.80 cfe/fsw/cfe-core/src/sb/ccsds.c File Reference

```
#include "common_types.h"
#include "ccsds.h"
#include "osapi.h"
#include "cfe_psp.h"
```

Functions

- void CCSDS_LoadCheckSum (CCSDS_CommandPacket_t *PktPtr)
- bool CCSDS_ValidCheckSum (CCSDS_CommandPacket_t *PktPtr)
- uint8 CCSDS_ComputeCheckSum (CCSDS_CommandPacket_t *PktPtr)

13.80.1 Function Documentation

13.80.1.1 CCSDS_ComputeCheckSum()

```
uint8 CCSDS_ComputeCheckSum (
            CCSDS_CommandPacket_t * PktPtr )
```

Definition at line 109 of file ccsds.c.

References CCSDS_RD_LEN, CCSDS_SpacePacket_t::Hdr, and CCSDS_CommandPacket_t::SpacePacket.

Referenced by CCSDS_LoadCheckSum(), and CCSDS_ValidCheckSum().

13.80.1.2 CCSDS_LoadCheckSum()

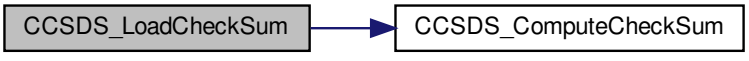
```
void CCSDS_LoadCheckSum (
            CCSDS_CommandPacket_t * PktPtr )
```

Definition at line 55 of file ccsds.c.

References CCSDS_ComputeCheckSum(), CCSDS_WR_CHECKSUM, and CCSDS_CommandPacket_t::Sec.

Referenced by CFE_SB_GenerateChecksum().

Here is the call graph for this function:



13.80.1.3 CCSDS_ValidCheckSum()

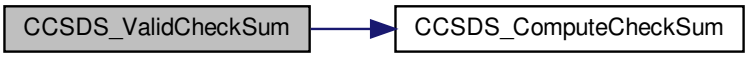
```
bool CCSDS_ValidCheckSum (
            CCSDS_CommandPacket_t * PktPtr )
```

Definition at line 85 of file ccsds.c.

References CCSDS_ComputeCheckSum().

Referenced by CFE_SB_ValidateChecksum().

Here is the call graph for this function:



13.81 cfe/fsw/cfe-core/src/sb/cfe_sb.mak File Reference

13.82 cfe/fsw/cfe-core/src/sb/cfe_sb_api.c File Reference

```
#include "common_types.h"
#include "private/cfe_private.h"
#include "cfe_sb_events.h"
#include "cfe_sb_priv.h"
#include "cfe_sb.h"
#include "osapi.h"
#include "cfe_es.h"
#include "cfe_psp.h"
#include "cfe_error.h"
#include <string.h>
```

Macros

- #define CFE_SB_TLM_PIPEDEPTHSTATS_SIZE (sizeof(CFE_SB.StatTlmMsg.Payload.PipeDepthStats) / sizeof(CFE_SB.StatTlmMsg.Payload.PipeDepthStats[0]))

Functions

- int32 CFE_SB_CreatePipe (CFE_SB_PipeId_t *PipeIdPtr, uint16 Depth, const char *PipeName)

  Creates a new software bus pipe.
- int32 CFE_SB_DeletePipe (CFE_SB_PipeId_t PipeId)

  Delete a software bus pipe.
- int32 CFE_SB_DeletePipeWithAppId (CFE_SB_PipeId_t PipeId, uint32 AppId)
- int32 CFE_SB_DeletePipeFull (CFE_SB_PipeId_t PipeId, uint32 AppId)
- int32 CFE_SB_SetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 Opts)

  Set options on a pipe.
- int32 CFE_SB_GetPipeOpts (CFE_SB_PipeId_t PipeId, uint8 *OptsPtr)

  Get options on a pipe.
- int32 CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_PipeId_t PipeId)

  Get the pipe name for a given id.
- int32 CFE_SB_GetPipeIdByName (CFE_SB_PipeId_t *PipeIdPtr, const char *PipeName)

  Get pipe id by pipe name.
- int32 CFE_SB_SubscribeEx (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, CFE_SB_Qos_t Quality, uint16 MsgLim)

  Subscribe to a message on the software bus.
- int32 CFE_SB_SubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, uint16 MsgLim)

  Subscribe to a message while keeping the request local to a cpu.
- int32 CFE_SB_Subscribe (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId)

  Subscribe to a message on the software bus with default parameters.
- int32 CFE_SB_SubscribeFull (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, CFE_SB_Qos_t Quality, uint16 MsgLim, uint8 Scope)
- int32 CFE_SB_Unsubscribe (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId)

  Remove a subscription to a message on the software bus.
- int32 CFE_SB_UnsubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId)

  Remove a subscription to a message on the software bus on the current CPU.
- int32 CFE_SB_UnsubscribeWithAppId (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, uint32 AppId)
- int32 CFE_SB_UnsubscribeFull (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, uint8 Scope, uint32 AppId)
- int32 CFE_SB_SendMsg (CFE_SB_Msg_t *MsgPtr)

  Send a software bus message.
- int32 CFE_SB_PassMsg (CFE_SB_Msg_t *MsgPtr)

  Passes a software bus message.
- int32 CFE_SB_SendMsgFull (CFE_SB_Msg_t *MsgPtr, uint32 TlmCntIncrements, uint32 CopyMode)
- int32 CFE_SB_RcvMsg (CFE_SB_MsgPtr_t *BufPtr, CFE_SB_PipeId_t PipeId, int32 TimeOut)

  Receive a message from a software bus pipe.
- uint32 CFE_SB_GetLastSenderId (CFE_SB_SenderId_t **Ptr, CFE_SB_PipeId_t PipeId)

  Retrieve the application Info of the sender for the last message.
- CFE_SB_Msg_t * CFE_SB_ZeroCopyGetPtr (uint16 MsgSize, CFE_SB_ZeroCopyHandle_t *BufferHandle)

  Get a buffer pointer to use for "zero copy" SB sends.
- int32 CFE_SB_ZeroCopyReleasePtr (CFE_SB_Msg_t *Ptr2Release, CFE_SB_ZeroCopyHandle_t BufferHandle)

  Release an unused "zero copy" buffer pointer.
- int32 CFE_SB_ZeroCopyReleaseDesc (CFE_SB_Msg_t *Ptr2Release, CFE_SB_ZeroCopyHandle_t BufferHandle)
- int32 CFE_SB_ZeroCopySend (CFE_SB_Msg_t *MsgPtr, CFE_SB_ZeroCopyHandle_t BufferHandle)

Send an SB message in "zero copy" mode.

- int32 CFE_SB_ZeroCopyPass (CFE_SB_Msg_t *MsgPtr, CFE_SB_ZeroCopyHandle_t BufferHandle)

  Pass an SB message in "zero copy" mode.

- int32 CFE_SB_ReadQueue (CFE_SB_PipeD_t *PipeDscPtr, uint32 TskId, CFE_SB_TimeOut_t Time_Out, CFE_SB_BufferD_t **Message)

13.82.1 Macro Definition Documentation

13.82.1.1 CFE_SB_TLM_PIPEDEPTHSTATS_SIZE

```
#define CFE_SB_TLM_PIPEDEPTHSTATS_SIZE (sizeof(CFE_SB.StatTlmMsg.Payload.PipeDepthStats) / sizeof(C↩
FE_SB.StatTlmMsg.Payload.PipeDepthStats[0]))
```

Definition at line 76 of file cfe_sb_api.c.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_RcvMsg(), and CFE_SB_SendMsgFull().

13.82.2 Function Documentation

13.82.2.1 CFE_SB_CreatePipe()

```
int32 CFE_SB_CreatePipe (
            CFE_SB_PipeId_t * PipeIdPtr,
            uint16 Depth,
            const char * PipeName )
```

Description

This routine creates and initializes an input pipe that the calling application can use to receive software bus messages. By default, no messages are routed to the new pipe. So, the application must use CFE_SB_Subscribe() to specify which messages it wants to receive on this pipe.

Assumptions, External Events, and Notes:

None

Parameters

in	*PipeIdPtr*	A pointer to a variable of type CFE_SB_PipeId_t, which will be filled in with the pipe ID information by the CFE_SB_CreatePipe routine.
in	*Depth*	The maximum number of messages that will be allowed on this pipe at one time.
in	*PipeName*	A string to be used to identify this pipe in error messages and routing information telemetry. The string must be no longer than OS_MAX_API_NAME. Longer strings will be truncated.
out	**PipeIdPtr*	The identifier for the created pipe.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MAX_PIPES_MET	This error code will be returned from CFE_SB_CreatePipe when the SB cannot accomodate the request to create a pipe because the maximum number of pipes (CFE_PLATFORM_SB_MAX_PIPES) are in use. This configuration parameter is defined in the cfe_platform_cfg.h file.
CFE_SB_PIPE_CR_ERR	The maximum number of queues(OS_MAX_QUEUES) are in use. Or possibly a lower level problem with creating the underlying queue has occurred such as a lack of memory. If the latter is the problem, the status code displayed in the event must be tracked.

Returns

See also

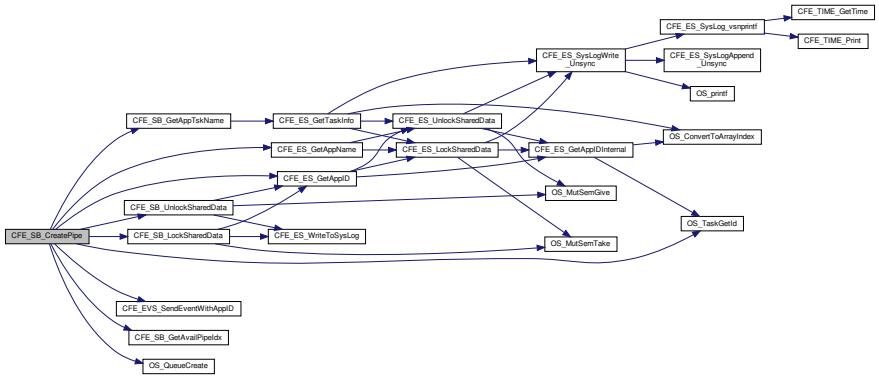
CFE_SB_DeletePipe CFE_SB_GetPipeOpts CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

Definition at line 104 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_SB_PipeD_t::AppName, CFE_ES_GetAppID(), CFE_ES_GetAppName(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_PIPE_DEPTH, CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_CR_PIPE_BAD_ARG_EID, CFE_SB_CR_PIPE_ERR_EID, CFE_SB_CR_PIPE_NAME_TAKEN_EID, CFE_SB_CR_PIPE_NO_FREE_EID, CFE_SB_GetAppTskName(), CFE_SB_GetAvailPipeIdx(), CFE_SB_IN_USE, CFE_SB_INVALID_PIPE, CFE_SB_LockSharedData(), CFE_SB_MAX_PIPES_MET, CFE_SB_MAX_PIPES_MET_EID, CFE_SB_PIPE_ADDED_EID, CFE_SB_PIPE_CR_ERR, CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter, CFE_SB_PipeD_t::CurrentBuff, CFE_SB_PipeDepthStats_t::Depth, cfe_sb_t::HKTlmMsg, CFE_SB_PipeD_t::InUse, CFE_SB_PipeDepthStats_t::InUse, NULL, OS_ERR_NAME_TAKEN, OS_ERR_NO_FREE_IDS, OS_MAX_API_NAME, OS_QueueCreate(), OS_SUCCESS, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_PipeDepthStats_t::PeakInUse, CFE_SB_StatsTlm_Payload_t::PeakPipesInUse, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_PipeD_t::PipeId, CFE_SB_PipeDepthStats_t::PipeId, CFE_SB_StatsTlm_Payload_t::PipesInUse, cfe_sb_t::PipeTbl, CFE_SB_PipeD_t::QueueDepth, CFE_SB_PipeD_t::SendErrors, cfe_sb_t::StatTlmMsg, CFE_SB_PipeD_t::SysQueueId, and CFE_SB_PipeD_t::ToTrashBuff.

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), and CFE_SB_AppInit().

Here is the call graph for this function:



13.82.2.2 CFE_SB_DeletePipe()

```
int32 CFE_SB_DeletePipe (
            CFE_SB_PipeId_t PipeId )
```

Description

This routine deletes an input pipe and cleans up all data structures associated with the pipe. All subscriptions made for this pipe by calls to CFE_SB_Subscribe will be automatically removed from the SB routing tables. Any messages in the pipe will be discarded.

Applications should not call this routine for all of their SB pipes as part of their orderly shutdown process, as the pipe will be deleted by the support framework at the appropriate time.

Assumptions, External Events, and Notes:

None

Parameters

in	*PipeId*	The pipe ID (obtained previously from CFE_SB_CreatePipe) of the pipe to be deleted.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.

Returns

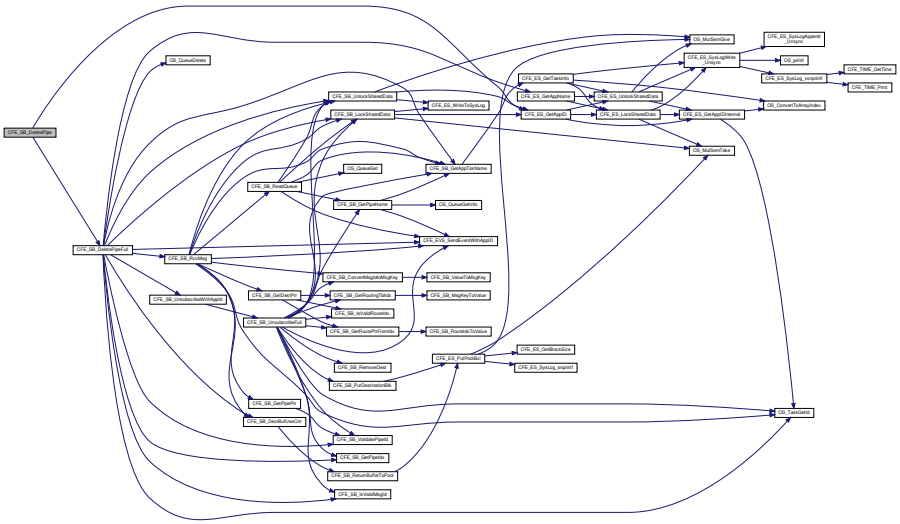
See also

CFE_SB_CreatePipe CFE_SB_GetPipeOpts CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

Definition at line 243 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), and CFE_SB_DeletePipeFull().

Here is the call graph for this function:



13.82.2.3 CFE_SB_DeletePipeFull()

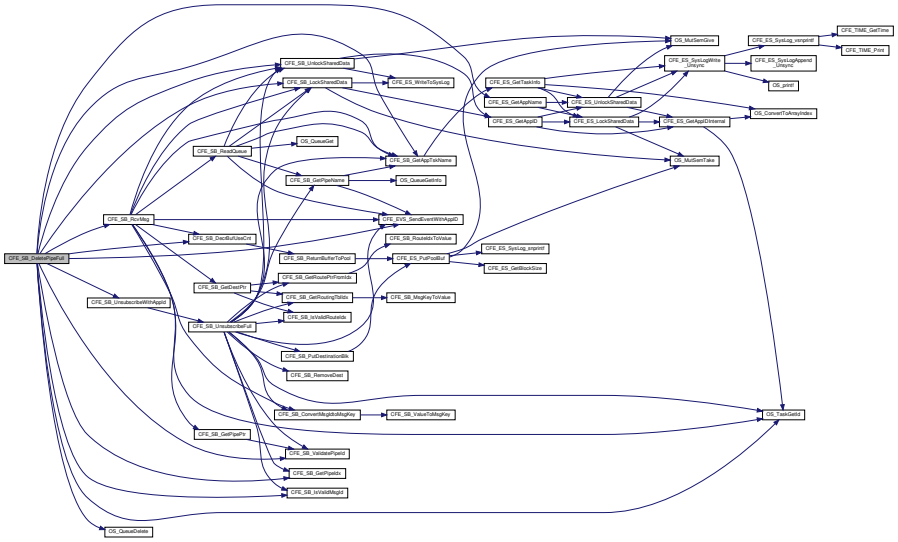
```
int32 CFE_SB_DeletePipeFull (
            CFE_SB_PipeId_t PipeId,
            uint32 AppId )
```

Definition at line 298 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppName(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_DecrBufUseCnt(), CFE_SB_DEL_PIPE_ERR1_EID, CFE_SB_DEL_PIPE_ERR2_EID, CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), CFE_SB_INVALID_PIPE, CFE_SB_IsValidMsgId(), CFE_SB_LockSharedData(), CFE_SB_NOT_IN_USE, CFE_SB_PIPE_DELETED_EID, CFE_SB_POLL, CFE_SB_RcvMsg(), CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SB_UnsubscribeWithAppId(), CFE_SB_UNUSED_QUEUE, CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter, CFE_SB_PipeD_t::CurrentBuff, CFE_SB_PipeDepthStats_t::Depth, cfe_sb_t::HKTlmMsg, CFE_SB_PipeD_t::InUse, CFE_SB_PipeDepthStats_t::InUse, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, NULL, OS_MAX_API_NAME, OS_QueueDelete(), OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_PipeDepthStats_t::PeakInUse, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_PipeD_t::PipeId, CFE_SB_PipeDepthStats_t::PipeId, CFE_SB_StatsTlm_Payload_t::PipesInUse, cfe_sb_t::PipeTbl, cfe_sb_t::RoutingTbl, cfe_sb_t::StatTlmMsg, CFE_SB_PipeD_t::SysQueueId, and CFE_SB_PipeD_t::ToTrashBuff.

Referenced by CFE_SB_DeletePipe(), and CFE_SB_DeletePipeWithAppId().

Here is the call graph for this function:



13.82.2.4 CFE_SB_DeletePipeWithAppId()

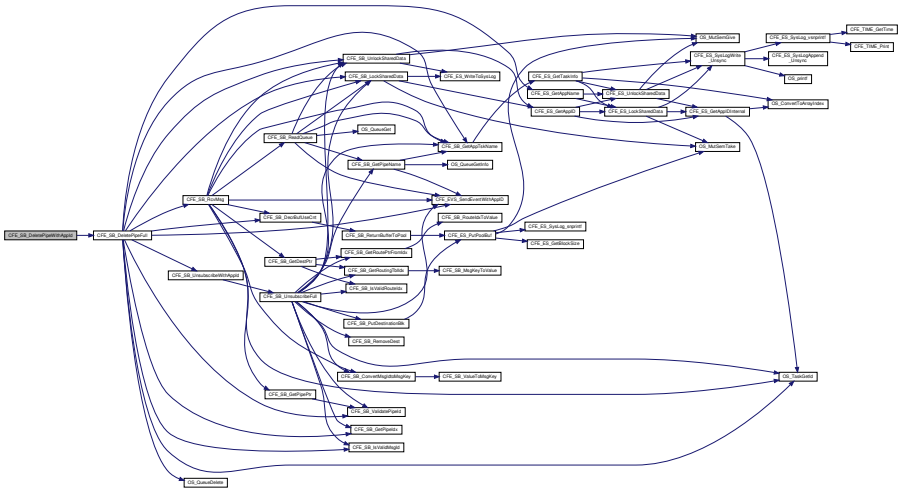
```
int32 CFE_SB_DeletePipeWithAppId (
            CFE_SB_PipeId_t PipeId,
            uint32 AppId )
```

Definition at line 271 of file cfe_sb_api.c.

References CFE_SB_DeletePipeFull().

Referenced by CFE_SB_CleanUpApp().

Here is the call graph for this function:



13.82.2.5 CFE_SB_GetLastSenderId()

```
uint32 CFE_SB_GetLastSenderId (
            CFE_SB_SenderId_t ** Ptr,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine can be used after a successful CFE_SB_RcvMsg call to find out which application sent the message that was received.

Assumptions, External Events, and Notes:

Note - If an error occurs in this API, the ∗Ptr value may be NULL or random. Therefore, it is recommended that the return code be tested for CFE_SUCCESS before reading the sender information.

Parameters

in	*Ptr*	A pointer to a local variable of type CFE_SB_SenderId_t. Typically a caller declares a ptr of type CFE_SB_SenderId_t (i.e. CFE_SB_SenderId_t ∗Ptr) then gives the address of that pointer (&Ptr) for this parameter. After a successful call to this API, ∗Ptr will point to the first byte of the CFE_SB_SenderId_t structure containing the sender information for the last message received on the given pipe. This should be used as a read-only pointer (in systems with an MMU, writes to this pointer may cause a memory protection fault). The ∗Ptr is valid only until the next call to CFE_SB_RcvMsg for the same pipe.
in	*PipeId*	The pipe ID of the pipe the message was taken from.

The sender's application ID	

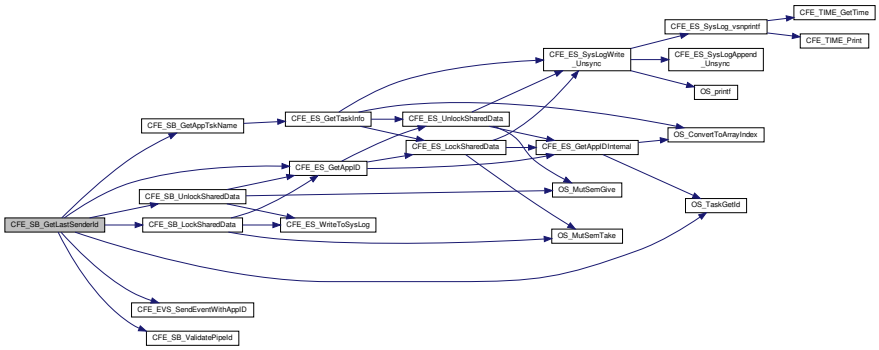
Returns

See also

Definition at line 1883 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GLS_INV_CALLER_EID, CFE_SB_LockSharedData(), CFE_SB_LSTSNDER_ERR1_EID, CFE_SB_LSTSNDER_ERR2_EID, CFE_SB_UnlockSharedData(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_PipeD_t::CurrentBuff, NULL, OS_MAX_API_NAME, OS_TaskGetId(), and cfe_sb_t::PipeTbl.

Here is the call graph for this function:



13.82.2.6 CFE_SB_GetPipeIdByName()

```
int32 CFE_SB_GetPipeIdByName (
            CFE_SB_PipeId_t * PipeIdPtr,
            const char * PipeName )
```

Description

This routine finds the pipe id for a pipe name.

Parameters

in	*PipeName*	The name of the pipe.
out	*PipeIdPtr*	The PipeId for that name.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_INVALID_PIPE	

Returns

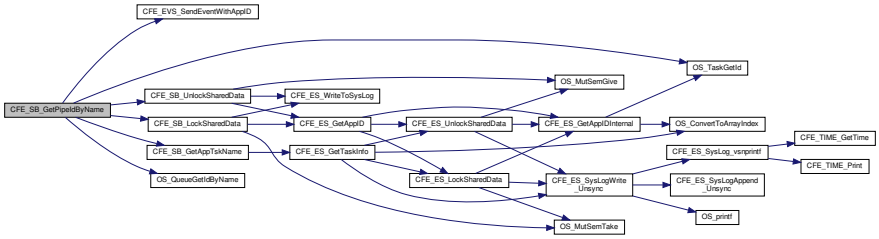
See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_PIPEOPTS_IGNOREMINE

Definition at line 628 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GETPIPEIDBYNAME_EID, CFE_SB_GETPIPEIDBYNAME_NAME_ERR_EID, CFE_SB_GETPIPEIDBYNAME_NULL_ERR_EID, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::GetPipeIdByNameErrorCounter, cfe_sb_t::HKTlmMsg, CFE_SB_PipeD_t::InUse, NULL, OS_MAX_API_NAME, OS_QueueGetIdByName(), OS_SUCCESS, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_PipeD_t::PipeId, cfe_sb_t::PipeTbl, and CFE_SB_PipeD_t::SysQueueId.

Here is the call graph for this function:



13.82.2.7 CFE_SB_GetPipeName()

```
int32 CFE_SB_GetPipeName (
            char * PipeNameBuf,
            size_t PipeNameSize,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine finds the pipe name for a pipe id.

Parameters

out	*PipeNameBuf*	The buffer to receive the pipe name.
in	*PipeNameSize*	The size (in chars) of the PipeName buffer.
in	*PipeId*	The PipeId for that name.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_INVALID_PIPE	

Returns

See also

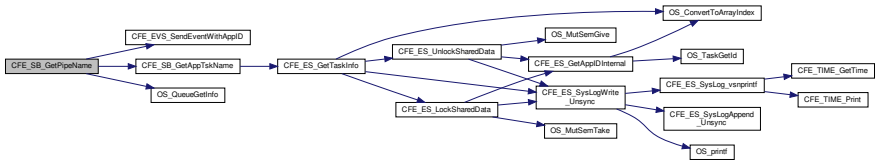
CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName

Definition at line 570 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GETPIPENAME_EID, CFE_SB_GETPIPENAME_ID_ERR_EID, CFE_SB_GETPIPENAME_NULL_PTR_EID, CFE_SUCCESS, OS_queue_prop_t::name, NULL, OS_MAX_API_NAME, OS_QueueGetInfo(), OS_SUCCESS, cfe_sb_t::PipeTbl, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SendRtgInfo(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.82.2.8 CFE_SB_GetPipeOpts()

```
int32 CFE_SB_GetPipeOpts (
            CFE_SB_PipeId_t PipeId,
            uint8 * OptPtr )
```

Description

This routine gets the current options on a pipe.

Parameters

in	*PipeId*	The pipe ID of the pipe to get options from.
out	*∗OptPtr*	A bit field of options.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.

Returns

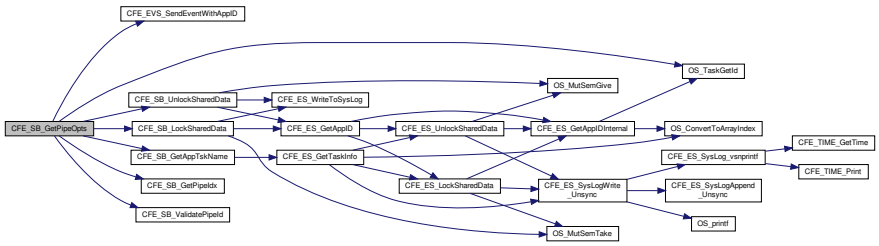
See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_SetPipeOpts CFE_SB_GetPipeIdByName CFE_SB_PIP↩ EOPTS_IGNOREMINE

Definition at line 509 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_Send↩ EventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), C↩ FE_SB_GETPIPEOPTS_EID, CFE_SB_GETPIPEOPTS_ID_ERR_EID, CFE_SB_GETPIPEOPTS_PTR_ERR_EID, CFE_SB_INVALID_PIPE, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, NULL, CFE_SB_PipeD_t::Opts, OS_MAX_API_NAME, OS_TaskGetId(), CF↩ E_SB_HousekeepingTlm_t::Payload, CFE_SB_HousekeepingTlm_Payload_t::PipeOptsErrorCounter, and cfe_sb_t::↩ PipeTbl.

Here is the call graph for this function:



13.82.2.9 CFE_SB_PassMsg()

```
int32 CFE_SB_PassMsg (
            CFE_SB_Msg_t * MsgPtr )
```

Description

This routine sends the specified message to all subscribers. The software bus will read the message ID from the message header to determine which pipes should receive the message. This routine is intended to pass messages not generated by the sending application.

Assumptions, External Events, and Notes:

- This routine will not normally wait for the receiver tasks to process the message before returning control to the caller's task.
- However, if a higher priority task is pending and subscribed to this message, that task may get to run before CFE_SB_PassMsg returns control to the caller.
- Unlike CFE_SB_SendMsg this routine will preserve the source sequence counter in a telemetry message.

Parameters

in	*MsgPtr*	A pointer to the message to be sent. This must point to the first byte of the software bus message header (CFE_SB_Msg_t).

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF↩ _MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

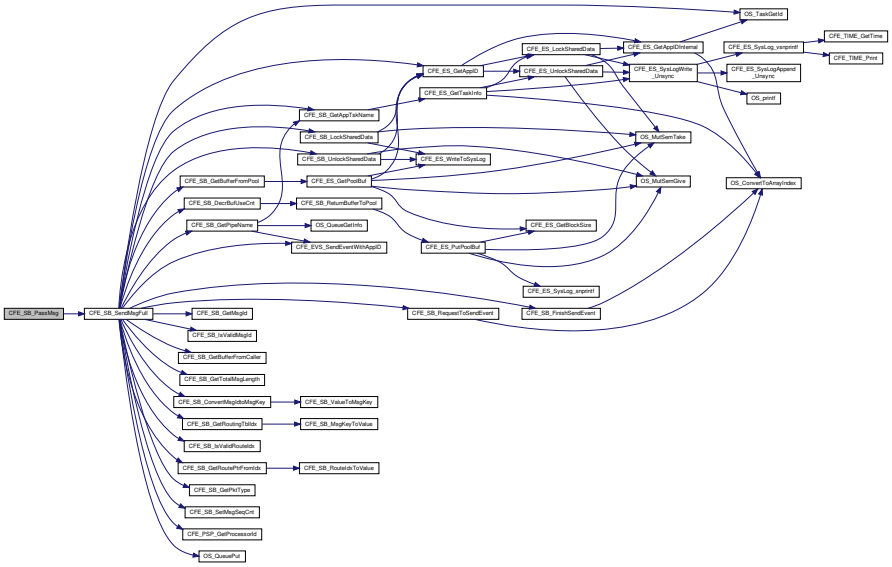
See also

CFE_SB_RcvMsg, CFE_SB_ZeroCopySend, CFE_SB_SendMsg

Definition at line 1353 of file cfe_sb_api.c.

References CFE_SB_DO_NOT_INCREMENT, CFE_SB_SEND_ONECOPY, and CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.82.2.10 CFE_SB_RcvMsg()

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

Description

This routine retrieves the next message from the specified pipe. If the pipe is empty, this routine will block until either a new message comes in or the timeout value is reached.

Assumptions, External Events, and Notes:

Note - If an error occurs in this API, the ∗BufPtr value may be NULL or random. Therefore, it is recommended that the return code be tested for CFE_SUCCESS before processing the message.

Parameters

in	*BufPtr*	A pointer to a local variable of type CFE_SB_MsgPtr_t. Typically a caller declares a ptr of type CFE_SB_Msg_t (i.e. CFE_SB_Msg_t ∗Ptr) then gives the address of that pointer (&Ptr) as this parmeter. After a successful receipt of a message, ∗BufPtr will point to the first byte of the software bus message header. This should be used as a read-only pointer (in systems with an MMU, writes to this pointer may cause a memory protection fault). The ∗BufPtr is valid only until the next call to CFE_SB_RcvMsg for the same pipe.
in	*PipeId*	The pipe ID of the pipe containing the message to be obtained.
in	*TimeOut*	The number of milliseconds to wait for a new message if the pipe is empty at the time of the call. This can also be set to CFE_SB_POLL for a non-blocking receive or CFE_SB_PEND_FOREVER to wait forever for a message to arrive.
out	∗*BufPtr*	A pointer to the message obtained from the pipe. Valid only until the next call to CFE_SB_RcvMsg for the same pipe.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_TIME_OUT	In CFE_SB_RcvMsg, this return value indicates that a packet has not been received in the time given in the "timeout" parameter.
CFE_SB_PIPE_RD_ERR	This return value indicates an error at the Queue read level. This error typically cannot be corrected by the caller. Some possible causes are: queue was not properly initialized or created, the number of bytes read from the queue was not the number of bytes requested in the read. The queue id is invalid. Similar errors regarding the pipe will be caught by higher level code in the Software Bus.
CFE_SB_NO_MESSAGE	When "Polling" a pipe for a message in CFE_SB_RcvMsg, this return value indicates that there was not a message on the pipe.

Returns

See also

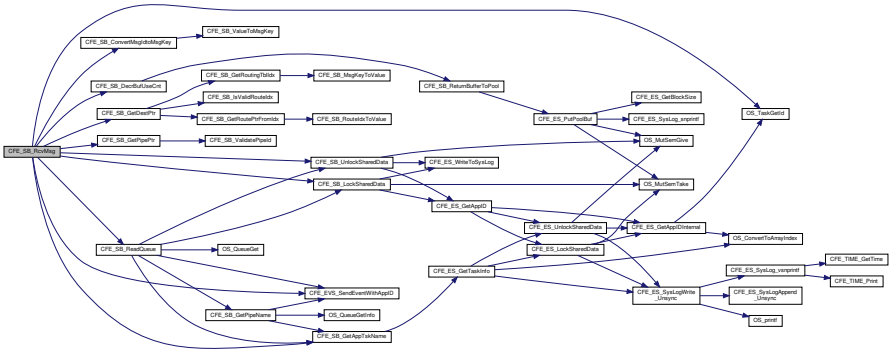
CFE_SB_SendMsg, CFE_SB_ZeroCopySend

Definition at line 1739 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_SB_DestinationD_t::BuffCount, CFE_SB_BufferD_t::Buffer, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_BAD_PIPEID_EID, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_DecrBufUseCnt(), CFE_SB_GetAppTskName(), CFE_SB_GetDestPtr(), CFE_SB_GetPipePtr(), CFE_SB_LockSharedData(), CFE_SB_RCV_BAD_ARG_EID, CFE_SB_ReadQueue(), CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_PipeD_t::CurrentBuff, cfe_sb_t::HKTlmMsg, CFE_SB_PipeDepthStats_t::InUse, CFE_SB_BufferD_t::MsgId, CFE_SB_HousekeepingTlm_Payload_t::MsgReceiveErrorCounter, NULL, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_PipeD_t::PipeId, cfe_sb_t::StatTlmMsg, and CFE_SB_PipeD_t::ToTrashBuff.

Referenced by CFE_ES_TaskMain(), CFE_EVS_TaskMain(), CFE_SB_DeletePipeFull(), and CFE_SB_TaskMain().

Here is the call graph for this function:



13.82.2.11 CFE_SB_ReadQueue()

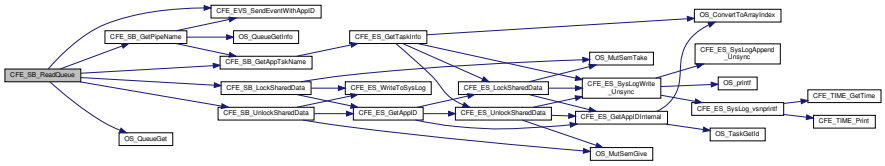
```
int32 CFE_SB_ReadQueue (
            CFE_SB_PipeD_t * PipeDscPtr,
            uint32 TskId,
            CFE_SB_TimeOut_t Time_Out,
            CFE_SB_BufferD_t ** Message )
```

Definition at line 2270 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_↩ SB_GetAppTskName(), CFE_SB_GetPipeName(), CFE_SB_LockSharedData(), CFE_SB_NO_MESSAGE, CFE_↩ SB_PEND_FOREVER, CFE_SB_PIPE_RD_ERR, CFE_SB_POLL, CFE_SB_Q_RD_ERR_EID, CFE_SB_TIME_O↩ UT, CFE_SB_UnlockSharedData(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::↩ InternalErrorCounter, OS_CHECK, OS_MAX_API_NAME, OS_PEND, OS_QUEUE_EMPTY, OS_QUEUE_TIMEOUT, OS_QueueGet(), OS_SUCCESS, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_PipeD_t::PipeId, and CFE_SB_↩ PipeD_t::SysQueueId.

Referenced by CFE_SB_RcvMsg().

Here is the call graph for this function:



13.82.2.12 CFE_SB_SendMsg()

```
int32 CFE_SB_SendMsg (
            CFE_SB_Msg_t * MsgPtr )
```

Description

This routine sends the specified message to all subscribers. The software bus will read the message ID from the message header to determine which pipes should receive the message.

Assumptions, External Events, and Notes:

- This routine will not normally wait for the receiver tasks to process the message before returning control to the caller's task.
- However, if a higher priority task is pending and subscribed to this message, that task may get to run before CFE_SB_SendMsg returns control to the caller.
- This function tracks and increments the source sequence counter of a telemetry message.

Parameters

in	*MsgPtr*	A pointer to the message to be sent. This must point to the first byte of the software bus message header (CFE_SB_Msg_t).

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF↩ _MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

See also

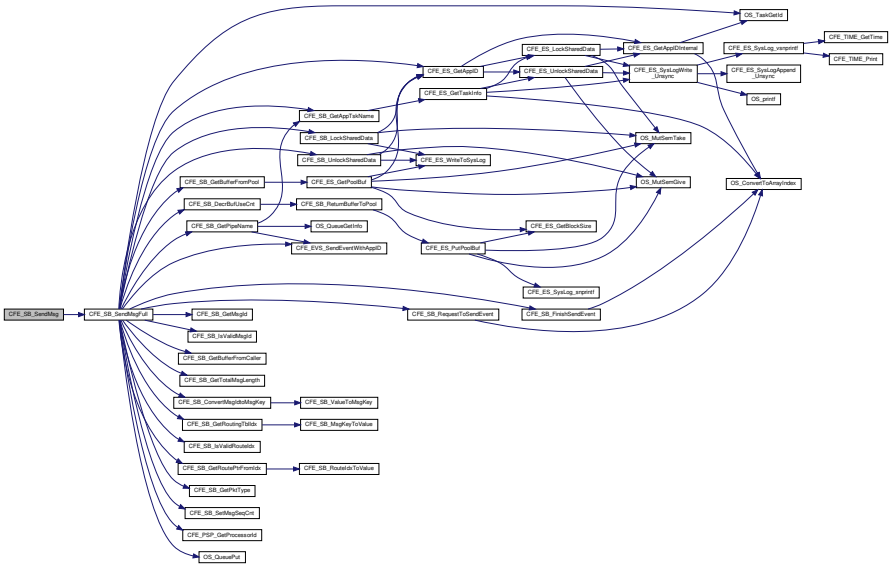
CFE_SB_RcvMsg, CFE_SB_ZeroCopySend, CFE_SB_PassMsg

Definition at line 1318 of file cfe_sb_api.c.

References CFE_SB_INCREMENT_TLM, CFE_SB_SEND_ONECOPY, and CFE_SB_SendMsgFull().

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_QueryOneCmd(), CFE_ES_SendMemPoolStatsCmd(), C↩ FE_ES_ShellOutputCommand(), CFE_EVS_ReportHousekeepingCmd(), CFE_SB_SendHKTlmCmd(), CFE_SB_↩ SendPrevSubsCmd(), CFE_SB_SendStatsCmd(), CFE_SB_SubscribeFull(), and EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.82.2.13 CFE_SB_SendMsgFull()

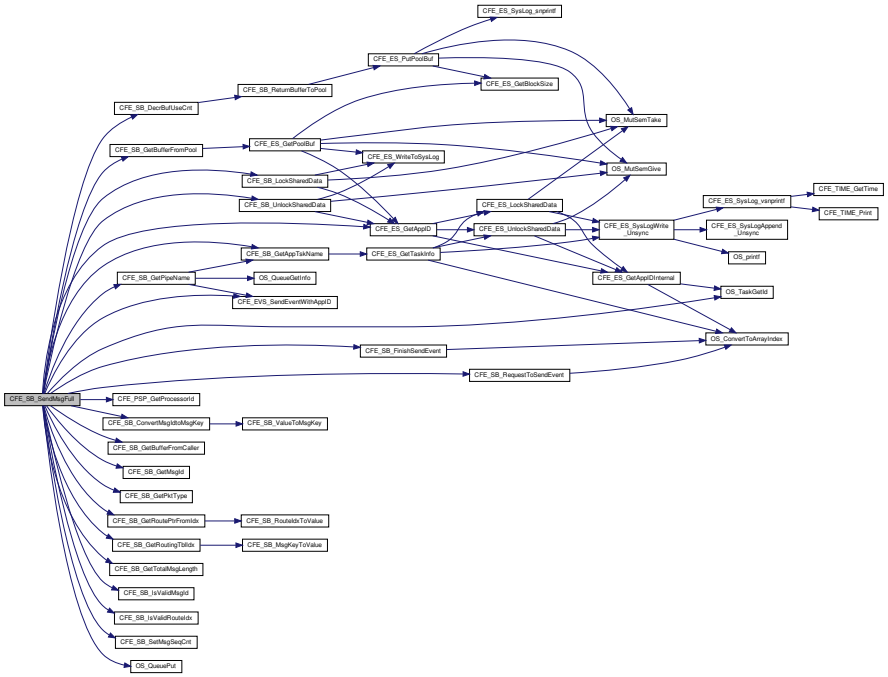
```
int32 CFE_SB_SendMsgFull (
              CFE_SB_Msg_t * MsgPtr,
              uint32 TlmCntIncrements,
              uint32 CopyMode )
```

Definition at line 1390 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_SB_SenderId_t::AppName, CFE_SB_BufferD_t::Buffer, CFE_ES_GetAppID(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEventWithAppID(), CFE_MISSION_SB_MAX_SB_MSG_SIZE, CFE_MISSION_SB_MSG_LIM_PERF_ID, CFE_MISSION_SB_PIPE_OFLOW_PERF_ID, CFE_PSP_GetProcessorId(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_BUF_ALOC_ERR, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_DecrBufUseCnt(), CFE_SB_FinishSendEvent(), CFE_SB_GET_BUF_ERR_EID, CFE_SB_GET_BUF_ERR_EID_BIT, CFE_SB_GetAppTskName(), CFE_SB_GetBufferFromCaller(), CFE_SB_GetBufferFromPool(), CFE_SB_GetMsgId(), CFE_SB_GetPipeName(), CFE_SB_GetPktType(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_GetTotalMsgLength(), CFE_SB_GRANTED, CFE_SB_INACTIVE, CFE_SB_INCREMENT_TLM, CFE_SB_IsValidMsgId(), CFE_SB_IsValidRouteIdx(), CFE_SB_LockSharedData(), CFE_SB_MSG_TOO_BIG, CFE_SB_MSG_TOO_BIG_EID, CFE_SB_MSGID_LIM_ERR_EID, CFE_SB_MSGID_LIM_ERR_EID_BIT, CFE_SB_PIPEOPTS_IGNOREMINE, CFE_SB_Q_FULL_ERR_EID, CFE_SB_Q_FULL_ERR_EID_BIT, CFE_SB_Q_WR_ERR_EID, CFE_SB_Q_WR_ERR_EID_BIT, CFE_SB_RequestToSendEvent(), CFE_SB_SEND_BAD_ARG_EID, CFE_SB_SEND_INV_MSGID_EID, CFE_SB_SEND_NO_SUBS_EID, CFE_SB_SEND_NO_SUBS_EID_BIT, CFE_SB_SEND_ZEROCOPY, CFE_SB_SetMsgSeqCnt(), CFE_SB_TLM, CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_EventBuf_t::EvtsToSnd, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::InternalErrorCounter, CFE_SB_PipeDepthStats_t::InUse, CFE_SB_RouteEntry_t::MsgId, CFE_SB_HousekeepingTlm_Payload_t::MsgLimitErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::MsgSendErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::NoSubscribersCounter, NULL, CFE_SB_PipeD_t::Opts, OS_MAX_API_NAME, OS_QUEUE_FULL, OS_QueuePut(), OS_SUCCESS, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_PipeDepthStats_t::PeakInUse, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_HousekeepingTlm_Payload_t::PipeOverflowErrorCounter, cfe_sb_t::PipeTbl, CFE_SB_SenderId_t::ProcessorId, CFE_SB_BufferD_t::Sender, cfe_sb_t::SenderReporting, CFE_SB_PipeD_t::SendErrors, CFE_SB_RouteEntry_t::SeqCnt, cfe_sb_t::StatTlmMsg, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_PassMsg(), CFE_SB_SendMsg(), CFE_SB_ZeroCopyPass(), and CFE_SB_ZeroCopySend().

Here is the call graph for this function:



13.82.2.14 CFE_SB_SetPipeOpts()

```
int32 CFE_SB_SetPipeOpts (
            CFE_SB_PipeId_t PipeId,
            uint8 Opts )
```

Description

This routine sets (or clears) options to alter the pipe's behavior. Options are (re)set every call to this routine.

Parameters

in	*PipeId*	The pipe ID of the pipe to set options on.
in	*Opts*	A bit field of options.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.

Returns

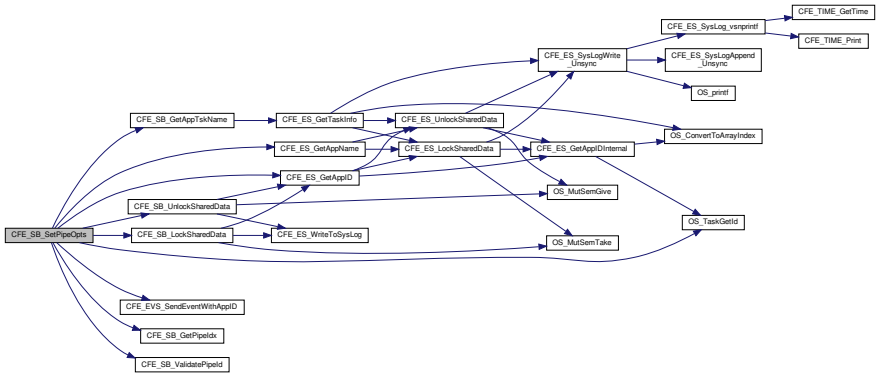
See also

CFE_SB_CreatePipe CFE_SB_DeletePipe CFE_SB_GetPipeOpts CFE_SB_GetPipeIdByName CFE_SB_PIP↩ EOPTS_IGNOREMINE

Definition at line 433 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_ES_GetAppName(), CFE_EVS_↩ EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_↩ ARGUMENT, CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), CFE_SB_INVALID_PIPE, CFE_SB_LockShared↩ Data(), CFE_SB_SETPIPEOPTS_EID, CFE_SB_SETPIPEOPTS_ID_ERR_EID, CFE_SB_SETPIPEOPTS_OWNE↩ R_ERR_EID, CFE_SB_UnlockSharedData(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, C↩ FE_SB_PipeD_t::Opts, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_↩ HousekeepingTlm_Payload_t::PipeOptsErrorCounter, and cfe_sb_t::PipeTbl.

Here is the call graph for this function:



13.82.2.15 CFE_SB_Subscribe()

```
int32 CFE_SB_Subscribe (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine adds the specified pipe to the destination list for the specified message ID. This is the same as CFE↩ _SB_SubscribeEx with the Quality field set to CFE_SB_Default_Qos and MsgLim set to CFE_PLATFORM_SB_↩ DEFAULT_MSG_LIMIT (4).

Assumptions, External Events, and Notes:

Note: As subscriptions are received, the destinations are added to the head of a linked list. During the sending of a message, the list is traversed beginning at the head of the list. Therefore the message will first be sent to the last subscriber. If an application has timing constraints and needs to receive a message in the shortest possible time, the developer may consider holding off its subscription until other applications have subscribed to the message.

Parameters

in	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_MAX_MSGS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.
CFE_SB_MAX_DESTS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER_PKT.
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

See also

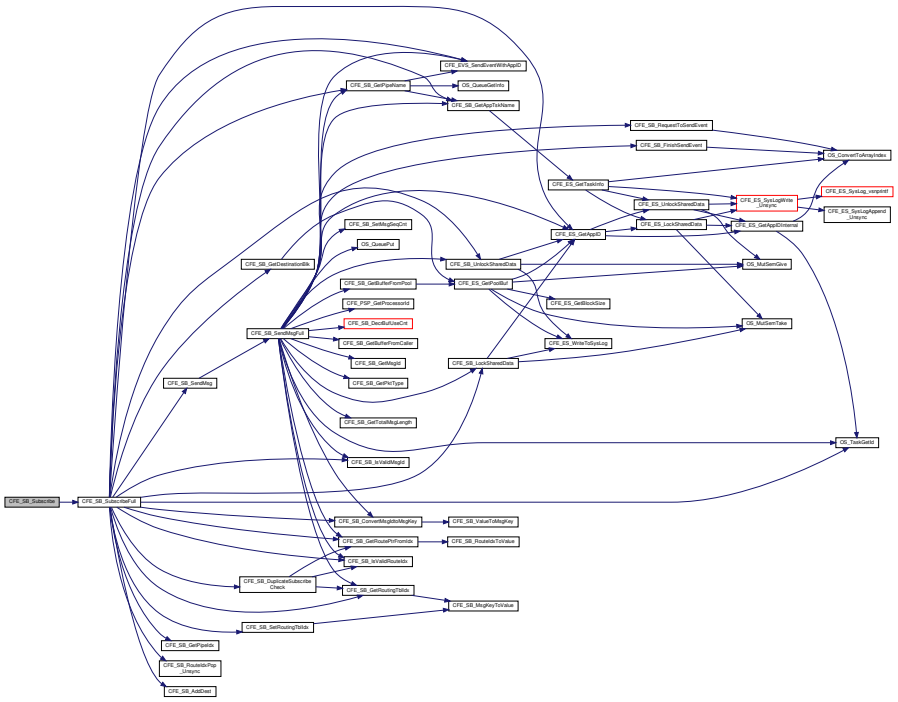
CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

Definition at line 828 of file cfe_sb_api.c.

References CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, CFE_SB_Default_Qos, CFE_SB_GLOBAL, and CFE_SB_SubscribeFull().

Referenced by CFE_SB_AppInit().

Here is the call graph for this function:



13.82.2.16 CFE_SB_SubscribeEx()

```
int32 CFE_SB_SubscribeEx (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            CFE_SB_Qos_t Quality,
            uint16 MsgLim )
```

Description

This routine adds the specified pipe to the destination list associated with the specified message ID.

Assumptions, External Events, and Notes:

Note: As subscriptions are received, the destinations are added to the head of a linked list. During the sending of a message, the list is traversed beginning at the head of the list. Therefore the message will first be sent to the last subscriber. If an application has timing constraints and needs to receive a message in the shortest possible time, the developer may consider holding off its subscription until other applications have subscribed to the message.

Parameters

in	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.
in	*Quality*	The requested Quality of Service (QoS) required of the messages. Most callers will use CFE_SB_Default_Qos for this parameter.
in	*MsgLim*	The maximum number of messages with this Message ID to allow in this pipe at the same time.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_MAX_MSGS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.
CFE_SB_MAX_DESTS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER_PKT.
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

See also

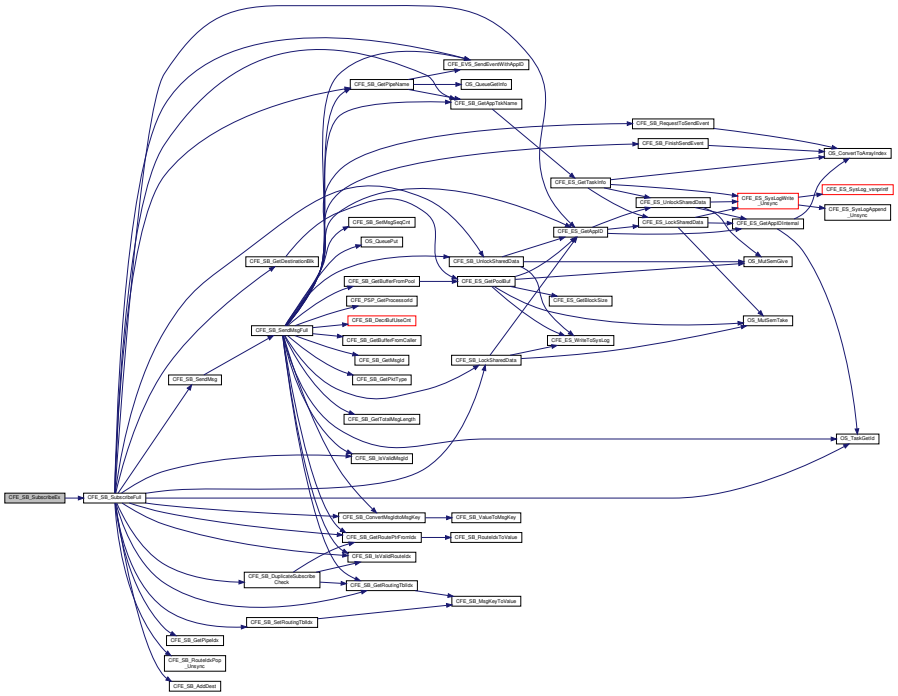
CFE_SB_Subscribe, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

Definition at line 750 of file cfe_sb_api.c.

References CFE_SB_GLOBAL, and CFE_SB_SubscribeFull().

Referenced by CFE_ES_TaskInit(), and CFE_EVS_TaskInit().

Here is the call graph for this function:



13.82.2.17 CFE_SB_SubscribeFull()

```
int32 CFE_SB_SubscribeFull (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            CFE_SB_Qos_t Quality,
            uint16 MsgLim,
            uint8 Scope )
```

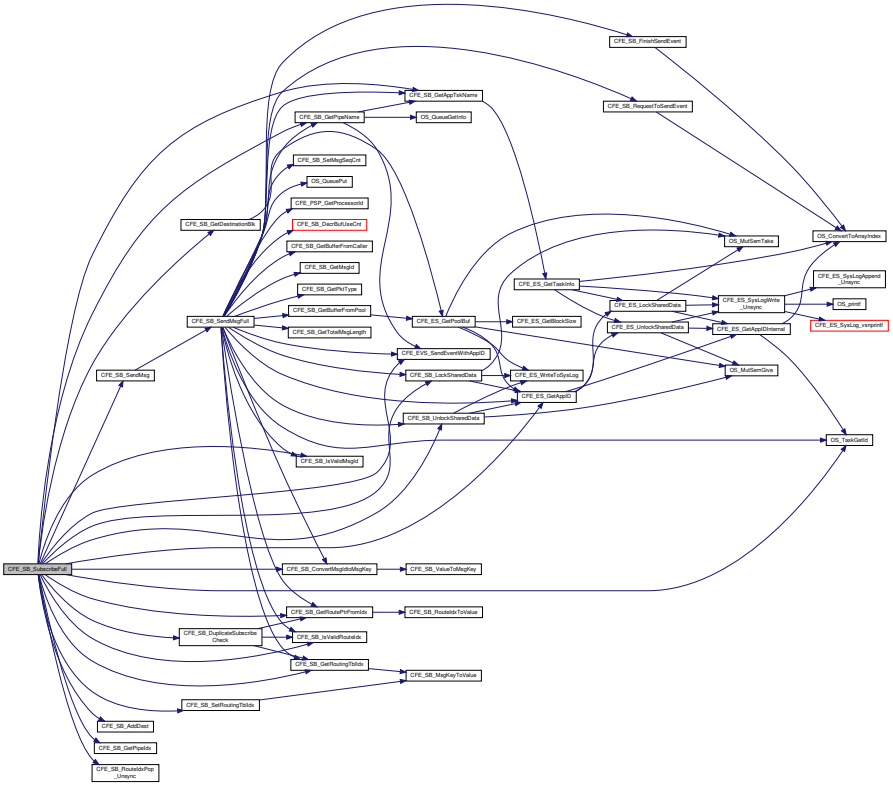
Definition at line 867 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_EVS_EventType_DEBUG, CFE↩
_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEventWithAppID(), CFE_PLAT↩
FORM_SB_MAX_DEST_PER_PKT, CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_ACTIVE, CFE_S↩
B_AddDest(), CFE_SB_BAD_ARGUMENT, CFE_SB_BUF_ALOC_ERR, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_↩
SB_DEST_BLK_ERR_EID, CFE_SB_DUP_SUBSCRIP_EID, CFE_SB_DUPLICATE, CFE_SB_DuplicateSubscribe↩
Check(), CFE_SB_ENABLE, CFE_SB_GetAppTskName(), CFE_SB_GetDestinationBlk(), CFE_SB_GetPipeIdx(),
CFE_SB_GetPipeName(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_GLOBAL, CFE↩
_SB_INVALID_PIPE, CFE_SB_IsValidMsgId(), CFE_SB_IsValidRouteIdx(), CFE_SB_LockSharedData(), CFE_SB_↩
MAX_DESTS_MET, CFE_SB_MAX_DESTS_MET_EID, CFE_SB_MAX_MSGS_MET, CFE_SB_MAX_MSGS_MET↩
_EID, CFE_SB_RouteIdxPop_Unsync(), CFE_SB_SendMsg(), CFE_SB_SetRoutingTblIdx(), CFE_SB_SUB_ARG_↩
ERR_EID, CFE_SB_SUB_INV_CALLER_EID, CFE_SB_SUB_INV_PIPE_EID, CFE_SB_SUBSCRIPTION, CFE_S↩
B_SUBSCRIPTION_RCVD_EID, CFE_SB_SUBSCRIPTION_RPT_EID, CFE_SB_UnlockSharedData(), CFE_SUC↩
CESS, CFE_SB_RouteEntry_t::Destinations, CFE_SB_HousekeepingTlm_Payload_t::DuplicateSubscriptionsCounter,

cfe_sb_t::HKTlmMsg, CFE_SB_RouteEntry_t::MsgId, CFE_SB_SingleSubscriptionTlm_Payload_t::MsgId, CFE_SB_StatsTlm_Payload_t::MsgIdsInUse, NULL, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_SingleSubscriptionTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMsgIdsInUse, CFE_SB_StatsTlm_Payload_t::PeakSubscriptionsInUse, CFE_SB_SingleSubscriptionTlm_Payload_t::Pipe, cfe_sb_t::PipeTbl, CFE_SB_Qos_t::Priority, CFE_SB_SingleSubscriptionTlm_Payload_t::Qos, CFE_SB_Qos_t::Reliability, cfe_sb_t::StatTlmMsg, cfe_sb_t::SubRprtMsg, CFE_SB_HousekeepingTlm_Payload_t::SubscribeErrorCounter, cfe_sb_t::SubscriptionReporting, CFE_SB_StatsTlm_Payload_t::SubscriptionsInUse, and CFE_SB_SingleSubscriptionTlm_Payload_t::SubType.

Referenced by CFE_SB_Subscribe(), CFE_SB_SubscribeEx(), and CFE_SB_SubscribeLocal().

Here is the call graph for this function:



13.82.2.18 CFE_SB_SubscribeLocal()

```
int32 CFE_SB_SubscribeLocal (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            uint16 MsgLim )
```

Description

This routine adds the specified pipe to the destination list for the specified message ID. This is similar to CFE_SB_SubscribeEx with the Quality field set to CFE_SB_Default_Qos and MsgLim set to CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT, but will not report the subscription. Subscription Reporting is enabled for interprocessor communication by way of the Software Bus Network (SBN) Application.

Assumptions, External Events, and Notes:

- This API is typically only used by Software Bus Network (SBN) Application

Parameters

in	*MsgId*	The message ID of the message to be subscribed to.
in	*PipeId*	The pipe ID of the pipe the subscribed message should be sent to.
in	*MsgLim*	The maximum number of messages with this Message ID to allow in this pipe at the same time.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_MAX_MSGS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another unique message ID because the platform configuration parameter CFE_PLATFORM_SB_MAX_MSG_IDS has been met.
CFE_SB_MAX_DESTS_MET	Will be returned when calling one of the SB subscription API's if the SB routing table cannot accomodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter CFE_PLATFORM_SB_MAX_DEST_PER↩_PKT.
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_S↩B_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Returns

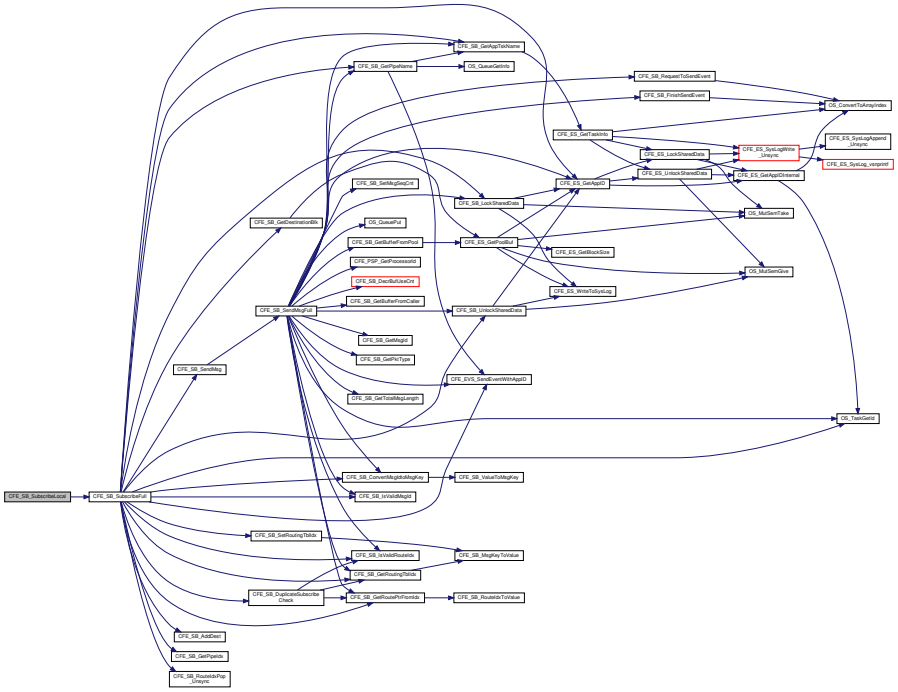
See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_Unsubscribe, CFE_SB_UnsubscribeLocal

Definition at line 790 of file cfe_sb_api.c.

References CFE_SB_Default_Qos, CFE_SB_LOCAL, and CFE_SB_SubscribeFull().

Here is the call graph for this function:



13.82.2.19 CFE_SB_Unsubscribe()

```
int32 CFE_SB_Unsubscribe (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine removes the specified pipe from the destination list for the specified message ID.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgId*	The message ID of the message to be unsubscribed.
in	*PipeId*	The pipe ID of the pipe the subscribed message should no longer be sent to.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_NO_SUBSCRIBERS	This error code is returned by the CFE_SB_Unsubscribe API if there has not been an entry in the routing tables for the MsgId/PipeId given as parameters.
CFE_SB_INTERNAL_ERR	This error code will be returned by the CFE_SB_Subscribe API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Returns

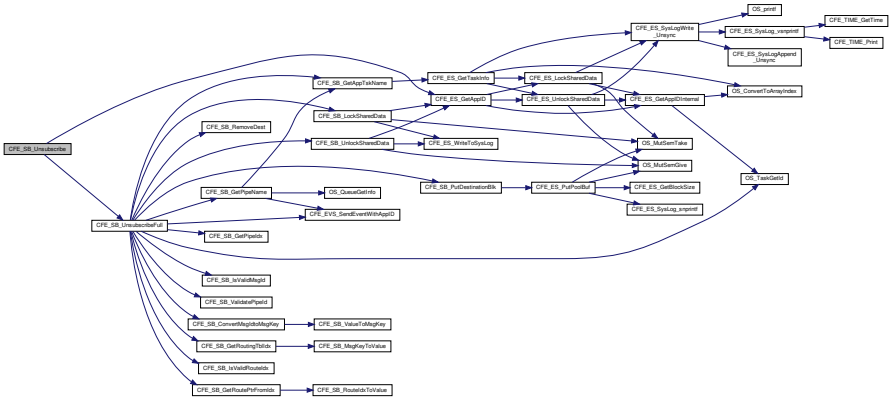
See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_UnsubscribeLocal

Definition at line 1075 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), CFE_SB_GLOBAL, and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.82.2.20 CFE_SB_UnsubscribeFull()

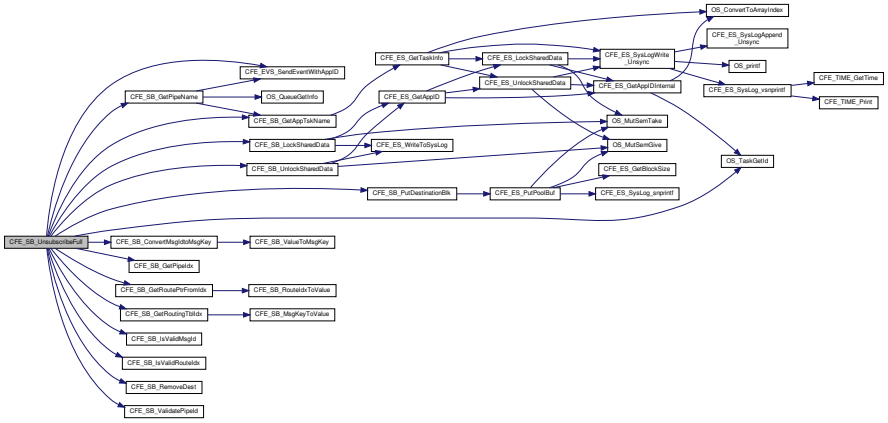
```
int32 CFE_SB_UnsubscribeFull (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            uint8 Scope,
            uint32 AppId )
```

Definition at line 1188 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ER↩ ROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_ARG↩ UMENT, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), CFE_SB_Get↩ PipeName(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_INVALID_PIPE, CFE_SB_↩ IsValidMsgId(), CFE_SB_IsValidRouteIdx(), CFE_SB_LockSharedData(), CFE_SB_PutDestinationBlk(), CFE_SB_↩ RemoveDest(), CFE_SB_SUBSCRIPTION_REMOVED_EID, CFE_SB_UnlockSharedData(), CFE_SB_UNSUB_AR↩ G_ERR_EID, CFE_SB_UNSUB_INV_CALLER_EID, CFE_SB_UNSUB_INV_PIPE_EID, CFE_SB_UNSUB_NO_SU↩ BS_EID, CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_RouteEntry_t::Destinations, CFE_SB_RouteEntry_↩ t::ListHeadPtr, CFE_SB_DestinationD_t::Next, NULL, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_StatsTlm↩ _t::Payload, CFE_SB_DestinationD_t::PipeId, cfe_sb_t::PipeTbl, cfe_sb_t::StatTlmMsg, and CFE_SB_StatsTlm_↩ Payload_t::SubscriptionsInUse.

Referenced by CFE_SB_Unsubscribe(), CFE_SB_UnsubscribeLocal(), and CFE_SB_UnsubscribeWithAppId().

Here is the call graph for this function:



13.82.2.21 CFE_SB_UnsubscribeLocal()

```
int32 CFE_SB_UnsubscribeLocal (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId )
```

Description

This routine removes the specified pipe from the destination list for the specified message ID on the current CPU.

Assumptions, External Events, and Notes:

- This API is typically only used by Software Bus Network (SBN) Application

Parameters

in	*MsgId*	The message ID of the message to be unsubscribed.
in	*PipeId*	The pipe ID of the pipe the subscribed message should no longer be sent to.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_NO_SUBSCRIBERS	This error code is returned by the CFE_SB_Unsubscribe API if there has not been an entry in the routing tables for the MsgId/PipeId given as parameters.
CFE_SB_INTERNAL_ERR	This error code will be returned by the CFE_SB_Subscribe API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Returns

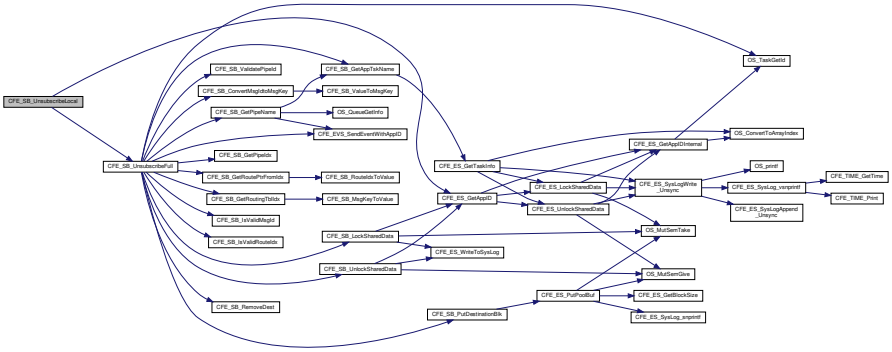
See also

CFE_SB_Subscribe, CFE_SB_SubscribeEx, CFE_SB_SubscribeLocal, CFE_SB_Unsubscribe

Definition at line 1110 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), CFE_SB_LOCAL, and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.82.2.22 CFE_SB_UnsubscribeWithAppId()

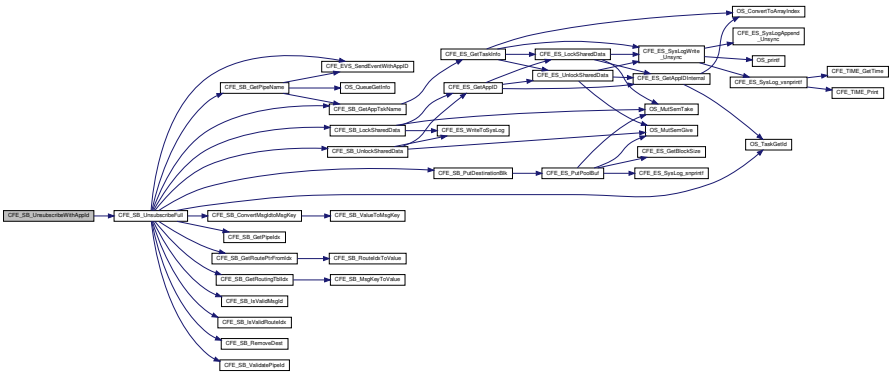
```
int32 CFE_SB_UnsubscribeWithAppId (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            uint32 AppId )
```

Definition at line 1147 of file cfe_sb_api.c.

References CFE_SB_LOCAL, and CFE_SB_UnsubscribeFull().

Referenced by CFE_SB_DeletePipeFull().

Here is the call graph for this function:



13.82.2.23 CFE_SB_ZeroCopyGetPtr()

```
CFE_SB_Msg_t* CFE_SB_ZeroCopyGetPtr (
            uint16 MsgSize,
            CFE_SB_ZeroCopyHandle_t * BufferHandle )
```

Description

This routine can be used to get a pointer to one of the software bus' internal memory buffers that are used for sending messages. The caller can use this memory buffer to build an SB message, then send it using the CFE_SB_ZeroCopySend function. This interface is more complicated than the normal CFE_SB_ZeroCopySend interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic.

Assumptions, External Events, and Notes:

1. The pointer returned by CFE_SB_ZeroCopyGetPtr is only good for one call to CFE_SB_ZeroCopySend.
2. Applications should be written as if CFE_SB_ZeroCopyGetPtr is equivalent to a `malloc()` and CFE_SB_ZeroCopySend is equivalent to a `free()`.
3. Applications must not de-reference the message pointer (for reading or writing) after the call to CFE_SB_ZeroCopySend.

Parameters

in	*MsgSize*	The size of the SB message buffer the caller wants (including the SB message header).
out	*BufferHandle*	A handle that must be supplied when sending or releasing in zero copy mode.

A pointer to a memory buffer that can be used to build one SB message for use with CFE_SB_ZeroCopySend.

Returns

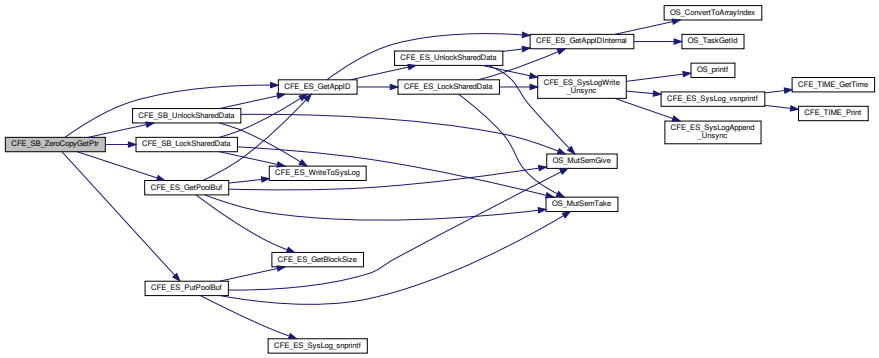
See also

CFE_SB_ZeroCopyReleasePtr, CFE_SB_ZeroCopySend

Definition at line 1962 of file cfe_sb_api.c.

References CFE_ES_GetAppID(), CFE_ES_GetPoolBuf(), CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, CFE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, CFE_SB_StatsTlm_Payload_t::PeakSBBuffersInUse, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, cfe_sb_t::StatTlmMsg, and cfe_sb_t::ZeroCopyTail.

Here is the call graph for this function:



13.82.2.24 CFE_SB_ZeroCopyPass()

```
int32 CFE_SB_ZeroCopyPass (
            CFE_SB_Msg_t * MsgPtr,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Description

This routine sends a message that has been created directly in an internal SB message buffer by an application (after a call to CFE_SB_ZeroCopyGetPtr). This interface is more complicated than the normal CFE_SB_SendMsg interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic. This version is intended to pass messages not generated by the caller (to preserve the source sequence count).

Assumptions, External Events, and Notes:

1. The pointer returned by CFE_SB_ZeroCopyGetPtr is only good for one call to CFE_SB_ZeroCopySend or CFE_SB_ZeroCopyPass.
2. Callers must not use the same SB message buffer for multiple sends.
3. Applications should be written as if CFE_SB_ZeroCopyGetPtr is equivalent to a `malloc()` and CFE_SB_ZeroCopyPass is equivalent to a `free()`.
4. Applications must not de-reference the message pointer (for reading or writing) after the call to CFE_SB_ZeroCopyPass.
5. Unlike CFE_SB_ZeroCopySend this routine will preserve the source sequence counter in a telemetry message.

Parameters

`in`	*MsgPtr*	A pointer to the SB message to be sent.
`in`	*BufferHandle*	The handle supplied with the CFE_SB_ZeroCopyGetPtr call.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF_MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.
CFE_SB_BUFFER_INVALID	This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Returns

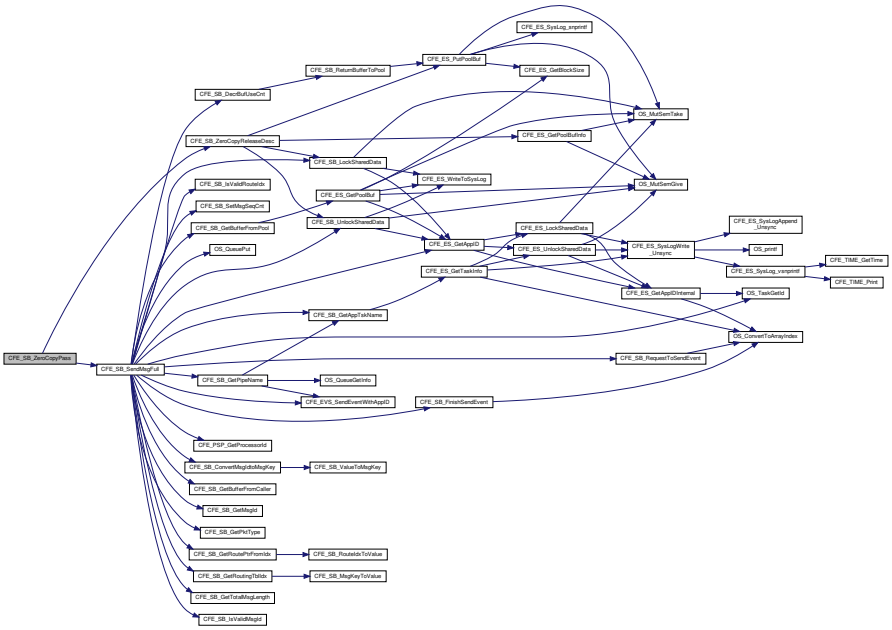
See also

CFE_SB_PassMsg, CFE_SB_ZeroCopySend, CFE_SB_ZeroCopyReleasePtr, CFE_SB_ZeroCopyGetPtr

Definition at line 2228 of file cfe_sb_api.c.

References CFE_SB_DO_NOT_INCREMENT, CFE_SB_SEND_ZEROCOPY, CFE_SB_SendMsgFull(), CFE_SB_↩ ZeroCopyReleaseDesc(), and CFE_SUCCESS.

Here is the call graph for this function:



13.82.2.25 CFE_SB_ZeroCopyReleaseDesc()

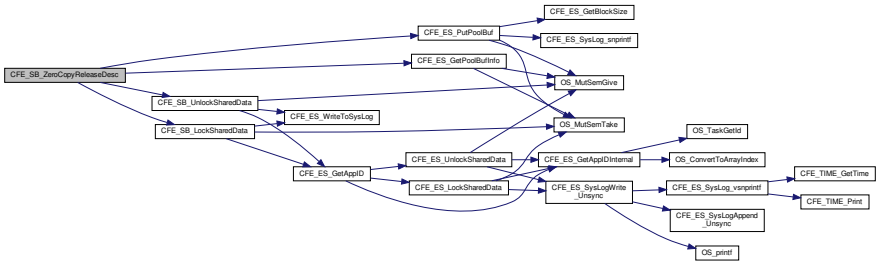
```
int32 CFE_SB_ZeroCopyReleaseDesc (
            CFE_SB_Msg_t * Ptr2Release,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Definition at line 2126 of file cfe_sb_api.c.

References CFE_ES_GetPoolBufInfo(), CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_BUFFER_INVALID, CFE_SB_↩ LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::↩ MemInUse, NULL, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, cfe_sb_t::StatTlmMsg, and cfe↩ _sb_t::ZeroCopyTail.

Referenced by CFE_SB_ZeroCopyPass(), CFE_SB_ZeroCopyReleasePtr(), and CFE_SB_ZeroCopySend().

Here is the call graph for this function:



13.82.2.26 CFE_SB_ZeroCopyReleasePtr()

```
int32 CFE_SB_ZeroCopyReleasePtr (
            CFE_SB_Msg_t * Ptr2Release,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Description

This routine can be used to release a pointer to one of the software bus' internal memory buffers.

Assumptions, External Events, and Notes:

1. This function is not needed for normal "zero copy" transfers. It is needed only for cleanup when an application gets a pointer using CFE_SB_ZeroCopyGetPtr, but (due to some error condition) never uses that pointer for a CFE_SB_ZeroCopySend

Parameters

in	*Ptr2Release*	A pointer to the SB internal buffer. This must be a pointer returned by a call to CFE_SB_ZeroCopyGetPtr, but never used in a call to CFE_SB_ZeroCopySend.
in	*BufferHandle*	This must be the handle supplied with the pointer when CFE_SB_ZeroCopyGetPtr was called.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BUFFER_INVALID	This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Returns

See also

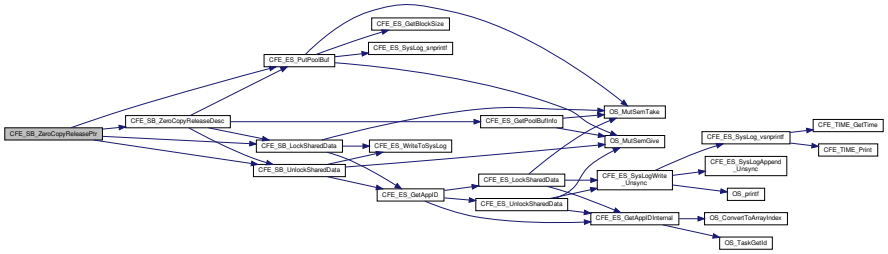
CFE_SB_ZeroCopyGetPtr, CFE_SB_ZeroCopySend

Definition at line 2074 of file cfe_sb_api.c.

References CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SB_ZeroCopyReleaseDesc(), CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_ZeroCopyReleaseAppId().

Here is the call graph for this function:



13.82.2.27 CFE_SB_ZeroCopySend()

```
int32 CFE_SB_ZeroCopySend (
            CFE_SB_Msg_t * MsgPtr,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Description

This routine sends a message that has been created directly in an internal SB message buffer by an application (after a call to CFE_SB_ZeroCopyGetPtr). This interface is more complicated than the normal CFE_SB_SendMsg interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic.

Assumptions, External Events, and Notes:

1. The pointer returned by CFE_SB_ZeroCopyGetPtr is only good for one call to CFE_SB_ZeroCopySend.
2. Callers must not use the same SB message buffer for multiple sends.
3. Applications should be written as if CFE_SB_ZeroCopyGetPtr is equivalent to a `malloc()` and CFE_SB_ZeroCopySend is equivalent to a `free()`.
4. Applications must not de-reference the message pointer (for reading or writing) after the call to CFE_SB_ZeroCopySend.
5. This function tracks and increments the source sequence counter of a telemetry message.

Parameters

in	*MsgPtr*	A pointer to the SB message to be sent.
in	*BufferHandle*	The handle supplied with the CFE_SB_ZeroCopyGetPtr call.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_BAD_ARGUMENT	A parameter given by a caller to a Software Bus API did not pass validation checks.
CFE_SB_MSG_TOO_BIG	The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter CFE_MISSION_SB_MAX_SB_MSG_SIZE in cfe_mission_cfg.h
CFE_SB_BUF_ALOC_ERR	This error code will be returned from CFE_SB_SendMsg when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter CFE_PLATFORM_SB_BUF↩ _MEMORY_BYTES specified in the cfe_platform_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.
CFE_SB_BUFFER_INVALID	This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Returns

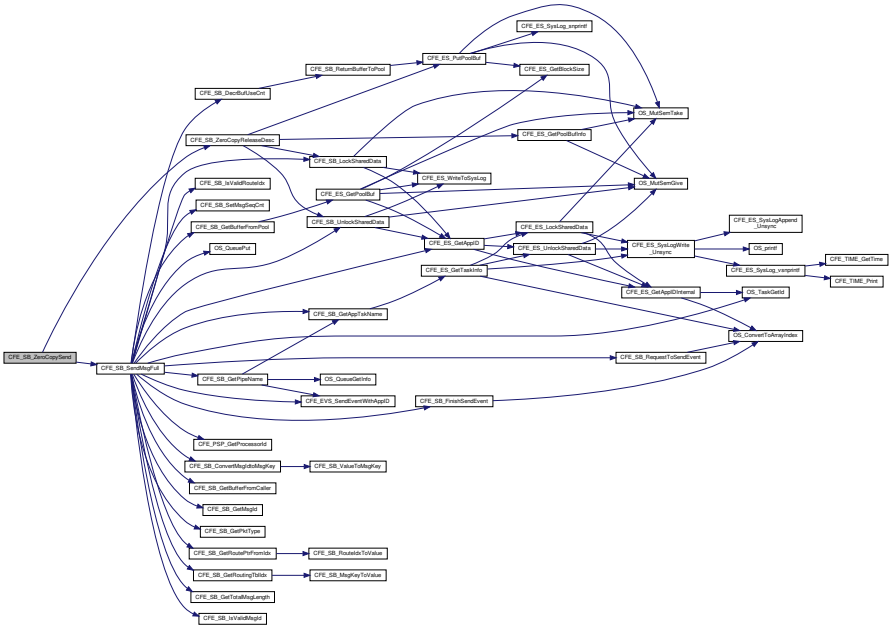
See also

CFE_SB_SendMsg, CFE_SB_RcvMsg, CFE_SB_ZeroCopyReleasePtr, CFE_SB_ZeroCopyGetPtr

Definition at line 2189 of file cfe_sb_api.c.

References CFE_SB_INCREMENT_TLM, CFE_SB_SEND_ZEROCOPY, CFE_SB_SendMsgFull(), CFE_SB_Zero↩ CopyReleaseDesc(), and CFE_SUCCESS.

Here is the call graph for this function:



13.83 cfe/fsw/cfe-core/src/sb/cfe_sb_buf.c File Reference

```
#include "cfe_sb_priv.h"
#include "cfe_sb_events.h"
#include "osapi.h"
#include "cfe_es.h"
#include "cfe_error.h"
```

Functions

- CFE_SB_BufferD_t ∗ CFE_SB_GetBufferFromPool (CFE_SB_MsgId_t MsgId, uint16 Size)
- CFE_SB_BufferD_t ∗ CFE_SB_GetBufferFromCaller (CFE_SB_MsgId_t MsgId, void ∗Address)
- int32 CFE_SB_ReturnBufferToPool (CFE_SB_BufferD_t ∗bd)
- int32 CFE_SB_DecrBufUseCnt (CFE_SB_BufferD_t ∗bd)
- CFE_SB_DestinationD_t ∗ CFE_SB_GetDestinationBlk (void)
- int32 CFE_SB_PutDestinationBlk (CFE_SB_DestinationD_t ∗Dest)

13.83.1 Function Documentation

13.83.1.1 CFE_SB_DecrBufUseCnt()

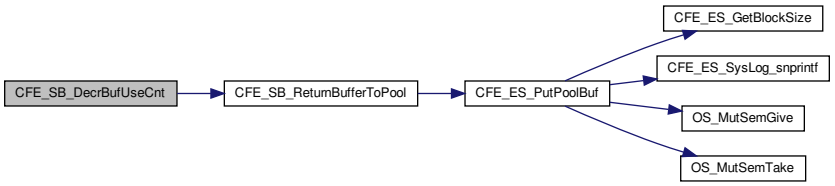
```
int32 CFE_SB_DecrBufUseCnt (
            CFE_SB_BufferD_t * bd )
```

Definition at line 178 of file cfe_sb_buf.c.

References CFE_SB_ReturnBufferToPool(), CFE_SUCCESS, and CFE_SB_BufferD_t::UseCount.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_RcvMsg(), and CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.83.1.2 CFE_SB_GetBufferFromCaller()

```
CFE_SB_BufferD_t* CFE_SB_GetBufferFromCaller (
            CFE_SB_MsgId_t MsgId,
            void * Address )
```

Definition at line 117 of file cfe_sb_buf.c.

References CFE_SB_BufferD_t::MsgId.

Referenced by CFE_SB_SendMsgFull().

13.83.1.3 CFE_SB_GetBufferFromPool()

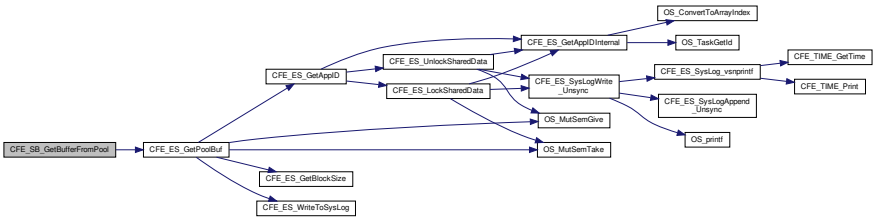
```
CFE_SB_BufferD_t* CFE_SB_GetBufferFromPool (
            CFE_SB_MsgId_t MsgId,
            uint16 Size )
```

Definition at line 60 of file cfe_sb_buf.c.

References CFE_ES_GetPoolBuf(), CFE_SB, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, CF↩E_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, CFE_SB_StatsTlm_Payload_t::PeakSB↩BuffersInUse, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, and cfe_sb_t::Stat↩TlmMsg.

Referenced by CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.83.1.4 CFE_SB_GetDestinationBlk()

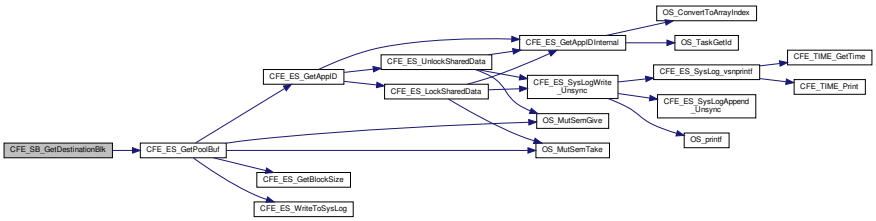
```
CFE_SB_DestinationD_t* CFE_SB_GetDestinationBlk (
            void  )
```

Definition at line 209 of file cfe_sb_buf.c.

References CFE_ES_GetPoolBuf(), CFE_SB, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, C↩FE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, CFE_SB_MemParams_t::PoolHdl, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_SubscribeFull().

Here is the call graph for this function:



13.83.1.5 CFE_SB_PutDestinationBlk()

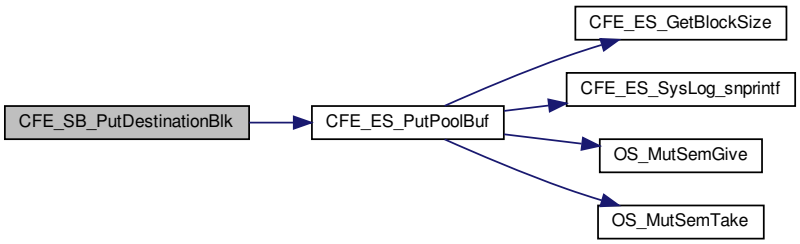
```
int32 CFE_SB_PutDestinationBlk (
            CFE_SB_DestinationD_t * Dest )
```

Definition at line 244 of file cfe_sb_buf.c.

References CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.83.1.6 CFE_SB_ReturnBufferToPool()

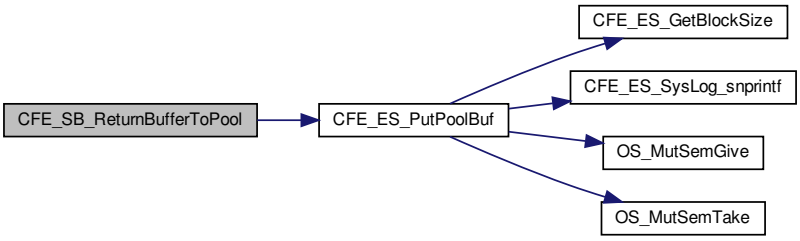
```
int32 CFE_SB_ReturnBufferToPool (
            CFE_SB_BufferD_t * bd )
```

Definition at line 143 of file cfe_sb_buf.c.

References CFE_ES_PutPoolBuf(), CFE_SB, CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_DecrBufUseCnt().

Here is the call graph for this function:



13.84 cfe/fsw/cfe-core/src/sb/cfe_sb_init.c File Reference

```
#include "cfe_sb_priv.h"
#include "cfe_sb.h"
#include "osapi.h"
#include "cfe_msgids.h"
#include "cfe_es.h"
#include "cfe_psp.h"
#include "cfe_error.h"
#include "cfe_sb_events.h"
#include <string.h>
```

Functions

- int32 CFE_SB_EarlyInit (void)

  Initializes the cFE core module API Library.
- int32 CFE_SB_InitBuffers (void)
- void CFE_SB_InitPipeTbl (void)
- void CFE_SB_InitMsgMap (void)
- void CFE_SB_InitRoutingTbl (void)

Variables

- uint32 CFE_SB_MemPoolDefSize [CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]

13.84.1 Function Documentation

13.84.1.1 CFE_SB_EarlyInit()

```
int32 CFE_SB_EarlyInit (
            void  )
```

Description

Initializes the cFE core module API Library

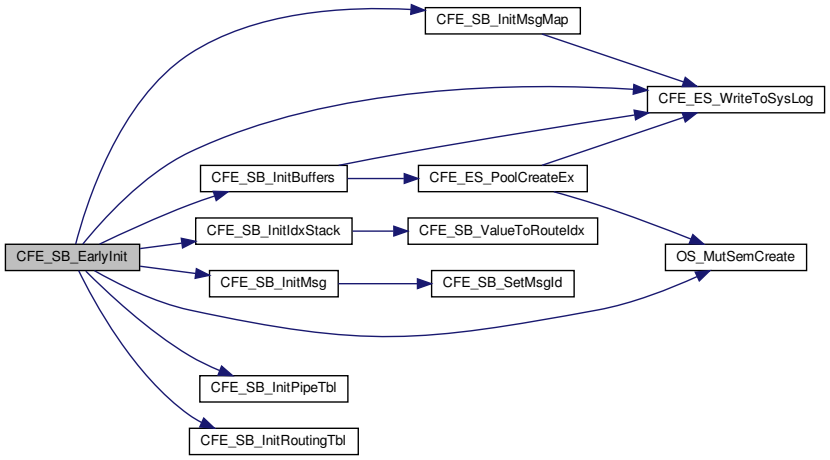
Assumptions, External Events, and Notes:

1. This function MUST be called before any module API's are called.

Definition at line 86 of file cfe_sb_init.c.

References CFE_ES_WriteToSysLog(), CFE_PLATFORM_SB_DEFAULT_REPORT_SENDER, CFE_SB, CFE_SB_Default_Qos, CFE_SB_DISABLE, CFE_SB_InitBuffers(), CFE_SB_InitIdxStack(), CFE_SB_InitMsg(), CFE_SB_InitMsgMap(), CFE_SB_InitPipeTbl(), CFE_SB_InitRoutingTbl(), CFE_SB_QOS_LOW_PRIORITY, CFE_SB_QOS_LOW_RELIABILITY, CFE_SB_STATS_TLM_MID, CFE_SUCCESS, NULL, OS_MutSemCreate(), OS_SUCCESS, CFE_SB_Qos_t::Priority, CFE_SB_Qos_t::Reliability, cfe_sb_t::SenderReporting, cfe_sb_t::SharedDataMutexId, cfe_sb_t::StatTlmMsg, cfe_sb_t::SubscriptionReporting, and cfe_sb_t::ZeroCopyTail.

Here is the call graph for this function:



13.84.1.2 CFE_SB_InitBuffers()

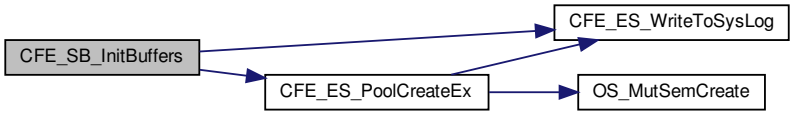
```
int32 CFE_SB_InitBuffers (
            void  )
```

Definition at line 152 of file cfe_sb_init.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_NO_MUTEX, CFE_ES_PoolCreateEx(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_SB_BUF_MEMORY_BYTES, CFE_SB, CFE_SB_MemPoolDefSize, CFE_SUCCESS, cfe_sb_t::Mem, and CFE_SB_MemParams_t::PoolHdl.

Referenced by CFE_SB_EarlyInit().

Here is the call graph for this function:



13.84.1.3 CFE_SB_InitMsgMap()

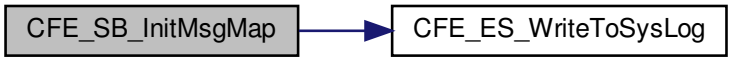
```
void CFE_SB_InitMsgMap (
             void  )
```

Definition at line 217 of file cfe_sb_init.c.

References CFE_ES_WriteToSysLog(), CFE_SB, CFE_SB_INVALID_ROUTE_IDX, CFE_SB_MAX_NUMBER_OF_↩ MSG_KEYS, and cfe_sb_t::MsgMap.

Referenced by CFE_SB_EarlyInit().

Here is the call graph for this function:



13.84.1.4 CFE_SB_InitPipeTbl()

```
void CFE_SB_InitPipeTbl (
             void  )
```

Definition at line 188 of file cfe_sb_init.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_INVALID_PIPE, CFE_SB_NOT_IN_USE, CFE↩ _SB_UNUSED_QUEUE, CFE_SB_PipeD_t::CurrentBuff, CFE_SB_PipeD_t::InUse, NULL, CFE_SB_PipeD_t::PipeId, cfe_sb_t::PipeTbl, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_EarlyInit().

13.84.1.5 CFE_SB_InitRoutingTbl()

```
void CFE_SB_InitRoutingTbl (
             void  )
```

Definition at line 252 of file cfe_sb_init.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_INVALID_MSG_ID, CFE_SB_RouteEntry_↩ t::Destinations, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, NULL, cfe_sb_t::RoutingTbl, and CFE_SB_RouteEntry_t::SeqCnt.

Referenced by CFE_SB_EarlyInit().

13.84.2 Variable Documentation

13.84.2.1 CFE_SB_MemPoolDefSize

```
uint32 CFE_SB_MemPoolDefSize[CFE_ES_MAX_MEMPOOL_BLOCK_SIZES]
```

Initial value:

```
=
{
    CFE_PLATFORM_SB_MAX_BLOCK_SIZE,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02,
    CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01
}
```

Definition at line 50 of file cfe_sb_init.c.

Referenced by CFE_SB_InitBuffers().

13.85 cfe/fsw/cfe-core/src/sb/cfe_sb_msg_id_util.c File Reference

```
#include "cfe_mission_cfg.h"
#include "ccsds.h"
#include "cfe_sb.h"
#include "osapi.h"
#include "cfe_error.h"
#include "cfe_sb_priv.h"
#include "cfe_sb_msg_id_util.h"
```

Functions

- CFE_SB_MsgKey_t CFE_SB_ConvertMsgIdtoMsgKey (CFE_SB_MsgId_t MsgId)
- CFE_SB_MsgId_t CFE_SB_GetMsgId (const CFE_SB_Msg_t *MsgPtr)

  Get the message ID of a software bus message.
- void CFE_SB_SetMsgId (CFE_SB_MsgPtr_t MsgPtr, CFE_SB_MsgId_t MsgId)

  Sets the message ID of a software bus message.

13.85.1 Function Documentation

13.85.1.1 CFE_SB_ConvertMsgIdtoMsgKey()

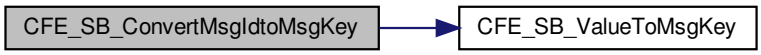
```
CFE_SB_MsgKey_t CFE_SB_ConvertMsgIdtoMsgKey (
            CFE_SB_MsgId_t MsgId )
```

Definition at line 111 of file cfe_sb_msg_id_util.c.

References CFE_SB_ValueToMsgKey().

Referenced by CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_RcvMsg(), CFE_SB_SendMsgFull(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.85.1.2 CFE_SB_GetMsgId()

```
CFE_SB_MsgId_t CFE_SB_GetMsgId (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the message ID from a software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message.

The software bus Message ID from the message header.

Returns

See also

CFE_SB_GetUserData, CFE_SB_SetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 132 of file cfe_sb_msg_id_util.c.

References CCSDS_CMD, CCSDS_RD_APID, CCSDS_RD_SID, CCSDS_RD_SUBSYSTEM_ID, CCSDS_RD_TYPE, CFE_SB_CMD_MESSAGE_TYPE, CFE_SB_Msg_t::Hdr, and CFE_SB_Msg_t::SpacePacket.

Referenced by CFE_ES_TaskPipe(), CFE_ES_VerifyCmdLength(), CFE_EVS_ProcessCommandPacket(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_VerifyCmdLength(), CFE_SB_ProcessCmdPipePkt(), CFE_SB_SendMsgFull(), and CFE_SB_VerifyCmdLength().

13.85.1.3 CFE_SB_SetMsgId()

```
void CFE_SB_SetMsgId (
             CFE_SB_MsgPtr_t MsgPtr,
             CFE_SB_MsgId_t MsgId )
```

Description

This routine sets the Message ID in a software bus message header.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*MsgId*	The message ID to put into the message header.

The software bus Message ID from the message header.

Returns

See also

CFE_SB_GetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 183 of file cfe_sb_msg_id_util.c.

References CCSDS_CLR_SEC_APIDQ, CCSDS_WR_APID, CCSDS_WR_EDS_VER, CCSDS_WR_ENDIAN, CCSDS_WR_PLAYBACK, CCSDS_WR_SID, CCSDS_WR_SUBSYSTEM_ID, CCSDS_WR_SYSTEM_ID, CCSDS_WR_TYPE, CCSDS_WR_VERS, CFE_PLATFORM_ENDIAN, CFE_SB_RD_APID_FROM_MSGID, CFE_SB_RD_SUBSYS_ID_FROM_MSGID, CFE_SB_RD_TYPE_FROM_MSGID, CFE_SPACECRAFT_ID, CFE_SB_Msg_t::Hdr, CCSDS_SpacePacket_t::Hdr, and CFE_SB_Msg_t::SpacePacket.

Referenced by CFE_SB_InitMsg().

13.86 cfe/fsw/cfe-core/src/sb/cfe_sb_msg_id_util.h File Reference

```
#include "common_types.h"
```

Macros

- #define CFE_SB_CMD_MESSAGE_TYPE 0x00000080 /* 1 bit (position 7) for Cmd/Tlm */
- #define CFE_SB_RD_APID_FROM_MSGID(MsgId) (MsgId & 0x0000007F) /* 0-6(7) bits for Pri Hdr APID */
- #define CFE_SB_RD_SUBSYS_ID_FROM_MSGID(MsgId) ((MsgId & 0x0000FF00) >> 8) /* bits 8-15(8) bits for APID Subsystem ID */
- #define CFE_SB_RD_TYPE_FROM_MSGID(MsgId) ((MsgId & CFE_SB_CMD_MESSAGE_TYPE) >> 7) /* 1 Cmd/Tlm Bit (bit #7) */

13.86.1 Macro Definition Documentation

13.86.1.1 CFE_SB_CMD_MESSAGE_TYPE

```
#define CFE_SB_CMD_MESSAGE_TYPE 0x00000080 /* 1 bit (position 7) for Cmd/Tlm */
```

For MESSAGE_FORMAT_IS_CCSDS_VER_2 the default layout of the message id is: 7 bits from the primary header APID 1 bit for the command/telemetry flag 0 bits from the Playback flag 8 bits from the secondary header APID qualifier (Subsystem) 0 bits from the secondary header APID qualifier as the System = 16 bits total

```
        Byte 1              Byte 0
 7 6 5 4 3 2 1 0     7      6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+|--------|+-+-+-+-+-+-+-+
| APID Qualifier |C/T flg | Pri Hdr APID |
+-+-+-+-+-+-+-+-+|--------|+-+-+-+-+-+-+-+
```

This layout may be modified via the 4 macros CFE_SB_CMD_MESSAGE_TYPE, CFE_SB_RD_APID_FROM_MSGID CFE_SB_RD_SUBSYS_ID_FROM_MSGID and CFE_SB_RD_TYPE_FROM_MSGID

Definition at line 68 of file cfe_sb_msg_id_util.h.

Referenced by CFE_SB_GetMsgId().

13.86.1.2 CFE_SB_RD_APID_FROM_MSGID

```
#define CFE_SB_RD_APID_FROM_MSGID(
            MsgId ) (MsgId & 0x0000007F) /* 0-6(7) bits for Pri Hdr APID */
```

Definition at line 73 of file cfe_sb_msg_id_util.h.

Referenced by CFE_SB_SetMsgId().

13.86.1.3 CFE_SB_RD_SUBSYS_ID_FROM_MSGID

```
#define CFE_SB_RD_SUBSYS_ID_FROM_MSGID(
            MsgId ) ( (MsgId & 0x0000FF00) >> 8) /* bits 8-15(8) bits for APID Subsystem ID */
```

Definition at line 74 of file cfe_sb_msg_id_util.h.

Referenced by CFE_SB_SetMsgId().

13.86.1.4 CFE_SB_RD_TYPE_FROM_MSGID

```
#define CFE_SB_RD_TYPE_FROM_MSGID(
            MsgId ) ( (MsgId & CFE_SB_CMD_MESSAGE_TYPE) >> 7) /* 1 Cmd/Tlm Bit (bit #7) */
```

Definition at line 75 of file cfe_sb_msg_id_util.h.

Referenced by CFE_SB_GetPktType(), and CFE_SB_SetMsgId().

13.87 cfe/fsw/cfe-core/src/sb/cfe_sb_priv.c File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "private/cfe_private.h"
#include "cfe_sb_priv.h"
#include "cfe_sb_msg_id_util.h"
#include "cfe_sb.h"
#include "ccsds.h"
#include "cfe_error.h"
#include "cfe_es.h"
#include <string.h>
```

Functions

- void CFE_SB_InitIdxStack (void)
- int32 CFE_SB_CleanUpApp (uint32 AppId)

  Removes SB resources associated with specified Application.
- CFE_SB_PipeId_t CFE_SB_GetAvailPipeIdx (void)
- CFE_SB_MsgRouteIdx_t CFE_SB_RouteIdxPop_Unsync (void)
- void CFE_SB_RouteIdxPush_Unsync (CFE_SB_MsgRouteIdx_t idx)
- uint8 CFE_SB_GetPipeIdx (CFE_SB_PipeId_t PipeId)
- void CFE_SB_LockSharedData (const char *FuncName, int32 LineNumber)
- void CFE_SB_UnlockSharedData (const char *FuncName, int32 LineNumber)
- CFE_SB_PipeD_t * CFE_SB_GetPipePtr (CFE_SB_PipeId_t PipeId)
- CFE_SB_DestinationD_t * CFE_SB_GetDestPtr (CFE_SB_MsgKey_t MsgKey, CFE_SB_PipeId_t PipeId)
- CFE_SB_MsgRouteIdx_t CFE_SB_GetRoutingTblIdx (CFE_SB_MsgKey_t MsgKey)
- void CFE_SB_SetRoutingTblIdx (CFE_SB_MsgKey_t MsgKey, CFE_SB_MsgRouteIdx_t Value)
- CFE_SB_RouteEntry_t * CFE_SB_GetRoutePtrFromIdx (CFE_SB_MsgRouteIdx_t RouteIdx)
- int32 CFE_SB_DuplicateSubscribeCheck (CFE_SB_MsgKey_t MsgKey, CFE_SB_PipeId_t PipeId)
- void CFE_SB_SetMsgSeqCnt (CFE_SB_MsgPtr_t MsgPtr, uint32 Count)
- int32 CFE_SB_ValidateMsgId (CFE_SB_MsgId_t MsgId)
- int32 CFE_SB_ValidatePipeId (CFE_SB_PipeId_t PipeId)
- char * CFE_SB_GetAppTskName (uint32 TaskId, char *FullName)
- uint8 CFE_SB_GetPktType (CFE_SB_MsgId_t MsgId)
- uint32 CFE_SB_RequestToSendEvent (uint32 TaskId, uint32 Bit)
- void CFE_SB_FinishSendEvent (uint32 TaskId, uint32 Bit)
- int32 CFE_SB_AddDest (CFE_SB_RouteEntry_t *RouteEntry, CFE_SB_DestinationD_t *NewNode)
- int32 CFE_SB_RemoveDest (CFE_SB_RouteEntry_t *RouteEntry, CFE_SB_DestinationD_t *NodeToRemove)
- int32 CFE_SB_ZeroCopyReleaseAppId (uint32 AppId)

13.87.1 Function Documentation

13.87.1.1 CFE_SB_AddDest()

```
int32 CFE_SB_AddDest (
            CFE_SB_RouteEntry_t * RouteEntry,
            CFE_SB_DestinationD_t * NewNode )
```

Definition at line 762 of file cfe_sb_priv.c.

References CFE_SUCCESS, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, NULL, and CFE↩
_SB_DestinationD_t::Prev.

Referenced by CFE_SB_SubscribeFull().

13.87.1.2 CFE_SB_CleanUpApp()

```
int32 CFE_SB_CleanUpApp (
            uint32 AppId )
```

Description

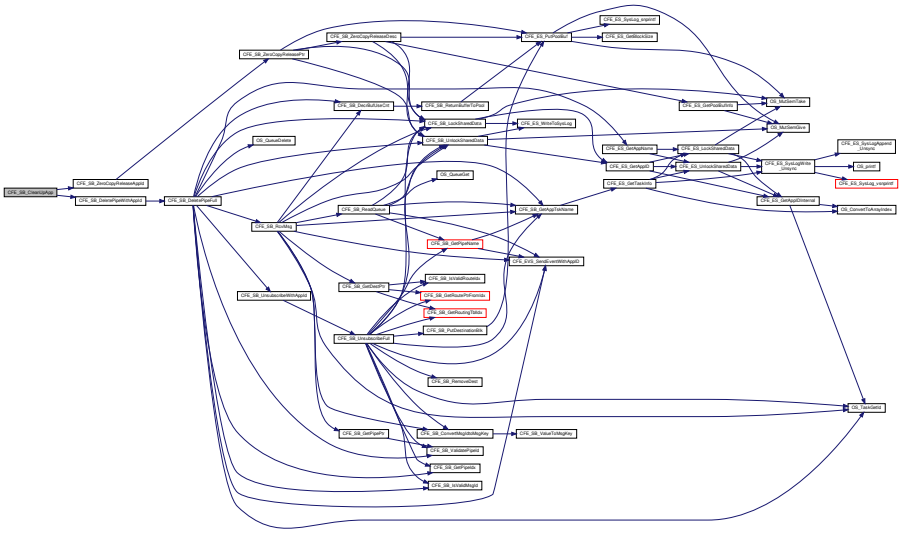
This function is called by cFE Executive Services to cleanup after an Application has been terminated. It frees resources that have been allocated to the specified Application.

Definition at line 126 of file cfe_sb_priv.c.

References CFE_SB_PipeD_t::AppId, CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_DeletePipeWithApp↩Id(), CFE_SB_IN_USE, CFE_SB_ZeroCopyReleaseAppId(), CFE_SUCCESS, CFE_SB_PipeD_t::InUse, CFE_SB_↩PipeD_t::PipeId, and cfe_sb_t::PipeTbl.

Referenced by CFE_ES_CleanUpApp().

Here is the call graph for this function:



13.87.1.3 CFE_SB_DuplicateSubscribeCheck()

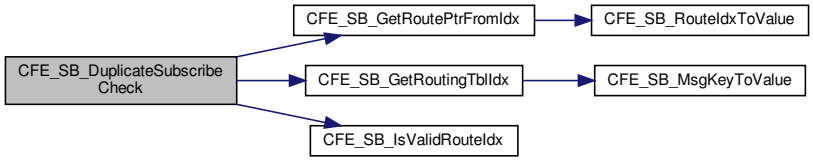
```
int32 CFE_SB_DuplicateSubscribeCheck (
            CFE_SB_MsgKey_t MsgKey,
            CFE_SB_PipeId_t PipeId )
```

Definition at line 505 of file cfe_sb_priv.c.

References CFE_SB_DUPLICATE, CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_IsValid↩RouteIdx(), CFE_SB_NO_DUPLICATE, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, and N↩ULL.

Referenced by CFE_SB_SubscribeFull().

Here is the call graph for this function:



13.87.1.4 CFE_SB_FinishSendEvent()

```
void CFE_SB_FinishSendEvent (
             uint32 TaskId,
             uint32 Bit )
```

Definition at line 739 of file cfe_sb_priv.c.

References CFE_CLR, CFE_SB, OS_ConvertToArrayIndex(), and cfe_sb_t::StopRecurseFlags.

Referenced by CFE_SB_SendMsgFull().

Here is the call graph for this function:

13.87.1.5 CFE_SB_GetAppTskName()

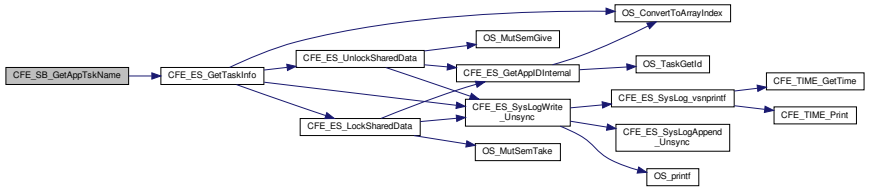
```
char* CFE_SB_GetAppTskName (
            uint32 TaskId,
            char * FullName )
```

Definition at line 627 of file cfe_sb_priv.c.

References CFE_ES_TaskInfo_t::AppName, CFE_ES_GetTaskInfo(), CFE_SUCCESS, OS_MAX_API_NAME, and CFE_ES_TaskInfo_t::TaskName.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.87.1.6 CFE_SB_GetAvailPipeIdx()

```
CFE_SB_PipeId_t CFE_SB_GetAvailPipeIdx (
            void  )
```

Definition at line 161 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_INVALID_PIPE, CFE_SB_NOT_IN_USE, CFE_SB_PipeD_t::InUse, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_CreatePipe().

13.87.1.7 CFE_SB_GetDestPtr()

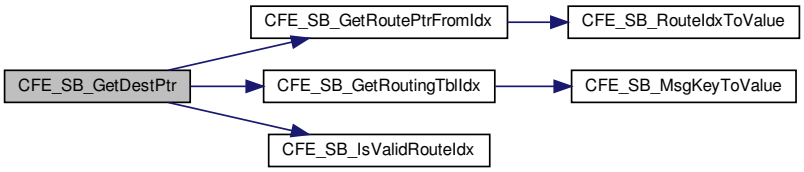
```
CFE_SB_DestinationD_t* CFE_SB_GetDestPtr (
            CFE_SB_MsgKey_t MsgKey,
            CFE_SB_PipeId_t PipeId )
```

Definition at line 384 of file cfe_sb_priv.c.

References CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_IsValidRouteIdx(), CFE_SB_↩ RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, and NULL.

Referenced by CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), and CFE_SB_RcvMsg().

Here is the call graph for this function:



13.87.1.8 CFE_SB_GetPipeIdx()

```
uint8 CFE_SB_GetPipeIdx (
            CFE_SB_PipeId_t PipeId )
```

Definition at line 250 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_INVALID_PIPE, CFE_SB_PipeD_t::InUse, CF↩ E_SB_PipeD_t::PipeId, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_GetPipeOpts(), CFE_SB_SetPipeOpts(), CFE_SB_Subscribe↩ Full(), and CFE_SB_UnsubscribeFull().

13.87.1.9 CFE_SB_GetPipePtr()

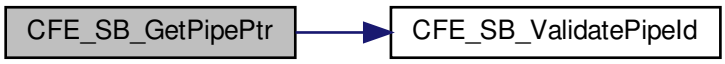
```
CFE_SB_PipeD_t* CFE_SB_GetPipePtr (
            CFE_SB_PipeId_t PipeId )
```

Definition at line 352 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_ValidatePipeId(), CFE_SUCCESS, NULL, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_RcvMsg(), and CFE_SB_SendRtgInfo().

Here is the call graph for this function:



13.87.1.10 CFE_SB_GetPktType()

```
uint8 CFE_SB_GetPktType (
            CFE_SB_MsgId_t MsgId )
```

Definition at line 675 of file cfe_sb_priv.c.

References CFE_SB_RD_TYPE_FROM_MSGID, and CFE_TST.

Referenced by CFE_SB_SendMsgFull().

13.87.1.11 CFE_SB_GetRoutePtrFromIdx()

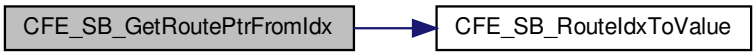
```
CFE_SB_RouteEntry_t* CFE_SB_GetRoutePtrFromIdx (
            CFE_SB_MsgRouteIdx_t RouteIdx )
```

Definition at line 486 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_RouteIdxToValue(), and cfe_sb_t::RoutingTbl.

Referenced by CFE_SB_DuplicateSubscribeCheck(), CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_GetDestPtr(), CFE↩
_SB_SendMapInfo(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRtgInfo(), CFE_SB_↩
SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.87.1.12 CFE_SB_GetRoutingTblIdx()

```
CFE_SB_MsgRouteIdx_t CFE_SB_GetRoutingTblIdx (
            CFE_SB_MsgKey_t MsgKey )
```

Definition at line 433 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_MsgKeyToValue(), and cfe_sb_t::MsgMap.

Referenced by CFE_SB_DuplicateSubscribeCheck(), CFE_SB_GetDestPtr(), CFE_SB_SendMapInfo(), CFE_SB_↩ SendMsgFull(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:

13.87.1.13 CFE_SB_InitIdxStack()

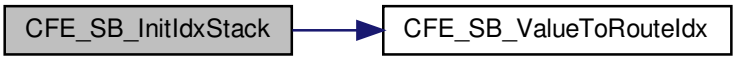
```
void CFE_SB_InitIdxStack (
            void  )
```

Definition at line 104 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_ValueToRouteIdx(), cfe_sb_t::RouteIdxStack, and cfe_sb_t::RouteIdxTop.

Referenced by CFE_SB_EarlyInit().

Here is the call graph for this function:



13.87.1.14 CFE_SB_LockSharedData()

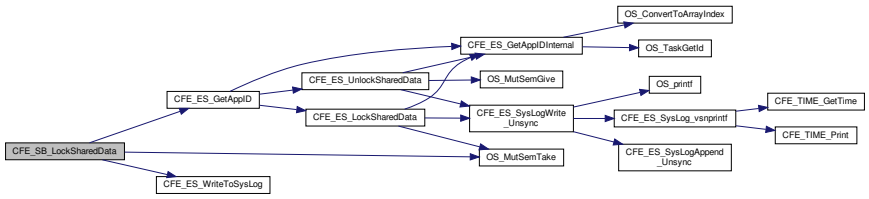
```
void CFE_SB_LockSharedData (
            const char * FuncName,
            int32 LineNumber )
```

Definition at line 282 of file cfe_sb_priv.c.

References CFE_ES_GetAppID(), CFE_ES_WriteToSysLog(), CFE_SB, OS_MutSemTake(), OS_SUCCESS, and cfe_sb_t::SharedDataMutexId.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

Here is the call graph for this function:



13.87.1.15 CFE_SB_RemoveDest()

```
int32 CFE_SB_RemoveDest (
            CFE_SB_RouteEntry_t * RouteEntry,
            CFE_SB_DestinationD_t * NodeToRemove )
```

Definition at line 810 of file cfe_sb_priv.c.

References CFE_SUCCESS, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, NULL, and CFE_SB_DestinationD_t::Prev.

Referenced by CFE_SB_UnsubscribeFull().

13.87.1.16 CFE_SB_RequestToSendEvent()

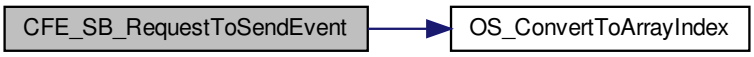
```
uint32 CFE_SB_RequestToSendEvent (
            uint32 TaskId,
            uint32 Bit )
```

Definition at line 707 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_DENIED, CFE_SB_GRANTED, CFE_SET, CFE_TST, OS_ConvertToArrayIndex(), and cfe_sb_t::StopRecurseFlags.

Referenced by CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.87.1.17 CFE_SB_RouteIdxPop_Unsync()

```
CFE_SB_MsgRouteIdx_t CFE_SB_RouteIdxPop_Unsync (
            void  )
```

Definition at line 195 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_INVALID_ROUTE_IDX, cfe_sb_t::RouteIdx↩Stack, and cfe_sb_t::RouteIdxTop.

Referenced by CFE_SB_SubscribeFull().

13.87.1.18 CFE_SB_RouteIdxPush_Unsync()

```
void CFE_SB_RouteIdxPush_Unsync (
            CFE_SB_MsgRouteIdx_t idx )
```

Definition at line 228 of file cfe_sb_priv.c.

References CFE_SB, cfe_sb_t::RouteIdxStack, and cfe_sb_t::RouteIdxTop.

13.87.1.19 CFE_SB_SetMsgSeqCnt()

```
void CFE_SB_SetMsgSeqCnt (
             CFE_SB_MsgPtr_t MsgPtr,
             uint32 Count )
```

Definition at line 552 of file cfe_sb_priv.c.

References CCSDS_WR_SEQ, and CFE_SB_Msg_t::Hdr.

Referenced by CFE_SB_SendMsgFull().

13.87.1.20 CFE_SB_SetRoutingTblIdx()

```
void CFE_SB_SetRoutingTblIdx (
             CFE_SB_MsgKey_t MsgKey,
             CFE_SB_MsgRouteIdx_t Value )
```

Definition at line 461 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_MsgKeyToValue(), and cfe_sb_t::MsgMap.

Referenced by CFE_SB_SubscribeFull().

Here is the call graph for this function:

13.87.1.21 CFE_SB_UnlockSharedData()

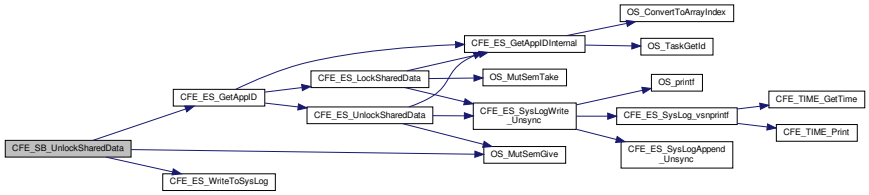
```
void CFE_SB_UnlockSharedData (
            const char * FuncName,
            int32 LineNumber )
```

Definition at line 317 of file cfe_sb_priv.c.

References CFE_ES_GetAppID(), CFE_ES_WriteToSysLog(), CFE_SB, OS_MutSemGive(), OS_SUCCESS, and cfe↩_sb_t::SharedDataMutexId.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipe↩IdByName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE↩_SB_SendPrevSubsCmd(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_S↩B_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

Here is the call graph for this function:



13.87.1.22 CFE_SB_ValidateMsgId()

```
int32 CFE_SB_ValidateMsgId (
            CFE_SB_MsgId_t MsgId )
```

Definition at line 572 of file cfe_sb_priv.c.

References CFE_SB_FAILED, CFE_SB_IsValidMsgId(), and CFE_SUCCESS.

Here is the call graph for this function:

13.87.1.23 CFE_SB_ValidatePipeId()

```
int32 CFE_SB_ValidatePipeId (
            CFE_SB_PipeId_t PipeId )
```

Definition at line 598 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_FAILED, CFE_SB_NOT_IN_USE, CFE_SUCC↩ ESS, CFE_SB_PipeD_t::InUse, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_S↩ B_GetLastSenderId(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_SetPipeOpts(), and CFE_SB_↩ UnsubscribeFull().

13.87.1.24 CFE_SB_ZeroCopyReleaseAppId()

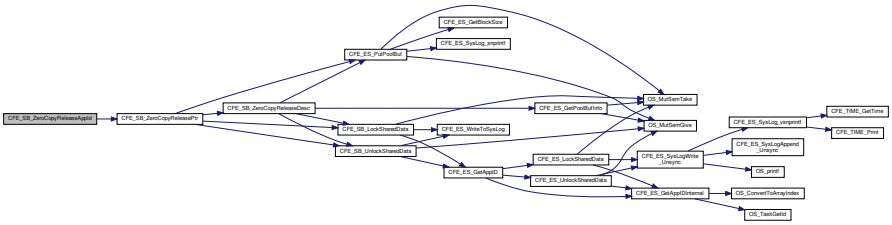
```
int32 CFE_SB_ZeroCopyReleaseAppId (
            uint32 AppId )
```

Definition at line 880 of file cfe_sb_priv.c.

References CFE_SB_ZeroCopyD_t::AppID, CFE_SB_ZeroCopyD_t::Buffer, CFE_SB, CFE_SB_ZeroCopyRelease↩ Ptr(), CFE_SUCCESS, NULL, CFE_SB_ZeroCopyD_t::Prev, and cfe_sb_t::ZeroCopyTail.

Referenced by CFE_SB_CleanUpApp().

Here is the call graph for this function:



13.88 cfe/fsw/cfe-core/src/sb/cfe_sb_priv.h File Reference

```
#include "common_types.h"
#include "private/cfe_private.h"
#include "cfe_sb.h"
#include "cfe_sb_msg.h"
#include "cfe_time.h"
#include "cfe_es.h"
```

Data Structures

- struct CFE_SB_MsgKey_t
- struct CFE_SB_MsgRouteIdx_t

  An wrapper for holding a routing table index.
- struct CFE_SB_BufferD_t
- struct CFE_SB_DestinationD_t
- struct CFE_SB_ZeroCopyD_t
- struct CFE_SB_RouteEntry_t
- struct CFE_SB_PipeD_t
- struct CFE_SB_MemParams_t
- struct cfe_sb_t
- struct CFE_SB_SendErrEventBuf_t
- struct CFE_SB_EventBuf_t

Macros

- #define CFE_SB_INVALID_ROUTE_IDX ((CFE_SB_MsgRouteIdx_t){ .RouteIdx = 0 })
- #define CFE_SB_INVALID_MSG_KEY ((CFE_SB_MsgKey_t){ .KeyIdx = 0 })
- #define CFE_SB_UNUSED_QUEUE 0xFFFF
- #define CFE_SB_INVALID_PIPE 0xFF
- #define CFE_SB_NO_DESTINATION 0xFF
- #define CFE_SB_FAILED 1
- #define SB_DONT_CARE 0
- #define CFE_SB_NO_DUPLICATE 0
- #define CFE_SB_DUPLICATE 1
- #define CFE_SB_INACTIVE 0
- #define CFE_SB_ACTIVE 1
- #define CFE_SB_GLOBAL 0
- #define CFE_SB_LOCAL 1
- #define CFE_SB_TLM 0
- #define CFE_SB_CMD 1
- #define CFE_SB_SEND_ZEROCOPY 0
- #define CFE_SB_SEND_ONECOPY 1
- #define CFE_SB_NOT_IN_USE 0
- #define CFE_SB_IN_USE 1
- #define CFE_SB_DISABLE 0
- #define CFE_SB_ENABLE 1
- #define CFE_SB_DENIED 0
- #define CFE_SB_GRANTED 1
- #define CFE_SB_DO_NOT_INCREMENT 0
- #define CFE_SB_INCREMENT_TLM 1
- #define CFE_SB_MAIN_LOOP_ERR_DLY 1000
- #define CFE_SB_CMD_PIPE_DEPTH 32
- #define CFE_SB_CMD_PIPE_NAME "SB_CMD_PIPE"
- #define CFE_SB_MAX_CFG_FILE_EVENTS_TO_FILTER 8
- #define CFE_SB_QOS_LOW_PRIORITY 0
- #define CFE_SB_QOS_LOW_RELIABILITY 0
- #define CFE_SB_PIPE_OVERFLOW (-1)
- #define CFE_SB_PIPE_WR_ERR (-2)

- #define CFE_SB_USECNT_ERR (-3)
- #define CFE_SB_FILE_IO_ERR (-5)
- #define CFE_SB_SEND_NO_SUBS_EID_BIT 0
- #define CFE_SB_GET_BUF_ERR_EID_BIT 1
- #define CFE_SB_MSGID_LIM_ERR_EID_BIT 2
- #define CFE_SB_Q_FULL_ERR_EID_BIT 3
- #define CFE_SB_Q_WR_ERR_EID_BIT 4
- #define CFE_SB_MAX_NUMBER_OF_MSG_KEYS (1+CFE_PLATFORM_SB_HIGHEST_VALID_MSGID)

Typedefs

- typedef uint16 CFE_SB_MsgKey_Atom_t

Functions

- int32 CFE_SB_AppInit (void)
- int32 CFE_SB_InitBuffers (void)
- void CFE_SB_InitPipeTbl (void)
- void CFE_SB_InitMsgMap (void)
- void CFE_SB_InitRoutingTbl (void)
- void CFE_SB_InitIdxStack (void)
- void CFE_SB_ResetCounts (void)
- void CFE_SB_RouteIdxPush_Unsync (CFE_SB_MsgRouteIdx_t idx)
- CFE_SB_MsgRouteIdx_t CFE_SB_RouteIdxPop_Unsync (void)
- CFE_SB_MsgKey_t CFE_SB_ConvertMsgIdtoMsgKey (CFE_SB_MsgId_t MsgId)
- void CFE_SB_LockSharedData (const char *FuncName, int32 LineNumber)
- void CFE_SB_UnlockSharedData (const char *FuncName, int32 LineNumber)
- void CFE_SB_ReleaseBuffer (CFE_SB_BufferD_t *bd, CFE_SB_DestinationD_t *dest)
- int32 CFE_SB_ReadQueue (CFE_SB_PipeD_t *PipeDscPtr, uint32 TskId, CFE_SB_TimeOut_t Time_Out, CFE_SB_BufferD_t **Message)
- int32 CFE_SB_WriteQueue (CFE_SB_PipeD_t *pd, uint32 TskId, const CFE_SB_BufferD_t *bd, CFE_SB_MsgId_t MsgId)
- CFE_SB_MsgRouteIdx_t CFE_SB_GetRoutingTblIdx (CFE_SB_MsgKey_t MsgKey)
- uint8 CFE_SB_GetPipeIdx (CFE_SB_PipeId_t PipeId)
- int32 CFE_SB_ReturnBufferToPool (CFE_SB_BufferD_t *bd)
- void CFE_SB_ProcessCmdPipePkt (void)
- int32 CFE_SB_DuplicateSubscribeCheck (CFE_SB_MsgKey_t MsgKey, CFE_SB_PipeId_t PipeId)
- void CFE_SB_SetRoutingTblIdx (CFE_SB_MsgKey_t MsgKey, CFE_SB_MsgRouteIdx_t Value)
- CFE_SB_RouteEntry_t * CFE_SB_GetRoutePtrFromIdx (CFE_SB_MsgRouteIdx_t RouteIdx)
- void CFE_SB_ResetCounters (void)
- void CFE_SB_SetMsgSeqCnt (CFE_SB_MsgPtr_t MsgPtr, uint32 Count)
- char * CFE_SB_GetAppTskName (uint32 TaskId, char *FullName)
- CFE_SB_BufferD_t * CFE_SB_GetBufferFromPool (CFE_SB_MsgId_t MsgId, uint16 Size)
- CFE_SB_BufferD_t * CFE_SB_GetBufferFromCaller (CFE_SB_MsgId_t MsgId, void *Address)
- CFE_SB_PipeD_t * CFE_SB_GetPipePtr (CFE_SB_PipeId_t PipeId)
- CFE_SB_PipeId_t CFE_SB_GetAvailPipeIdx (void)
- CFE_SB_DestinationD_t * CFE_SB_GetDestPtr (CFE_SB_MsgKey_t MsgKey, CFE_SB_PipeId_t PipeId)
- int32 CFE_SB_DeletePipeWithAppId (CFE_SB_PipeId_t PipeId, uint32 AppId)
- int32 CFE_SB_DeletePipeFull (CFE_SB_PipeId_t PipeId, uint32 AppId)

- int32 CFE_SB_SubscribeFull (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, CFE_SB_Qos_t Quality, uint16 MsgLim, uint8 Scope)
- int32 CFE_SB_UnsubscribeWithAppId (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, uint32 AppId)
- int32 CFE_SB_UnsubscribeFull (CFE_SB_MsgId_t MsgId, CFE_SB_PipeId_t PipeId, uint8 Scope, uint32 AppId)
- int32 CFE_SB_SendMsgFull (CFE_SB_Msg_t *MsgPtr, uint32 TlmCntIncrements, uint32 CopyMode)
- int32 CFE_SB_SendRtgInfo (const char *Filename)
- int32 CFE_SB_SendPipeInfo (const char *Filename)
- int32 CFE_SB_SendMapInfo (const char *Filename)
- int32 CFE_SB_ZeroCopyReleaseDesc (CFE_SB_Msg_t *Ptr2Release, CFE_SB_ZeroCopyHandle_t BufferHandle)
- int32 CFE_SB_ZeroCopyReleaseAppId (uint32 AppId)
- int32 CFE_SB_DecrBufUseCnt (CFE_SB_BufferD_t *bd)
- int32 CFE_SB_ValidateMsgId (CFE_SB_MsgId_t MsgId)
- int32 CFE_SB_ValidatePipeId (CFE_SB_PipeId_t PipeId)
- uint8 CFE_SB_GetPktType (CFE_SB_MsgId_t MsgId)
- void CFE_SB_IncrCmdCtr (int32 status)
- void CFE_SB_FileWriteByteCntErr (const char *Filename, uint32 Requested, uint32 Actual)
- void CFE_SB_SetSubscriptionReporting (uint32 state)
- uint32 CFE_SB_FindGlobalMsgIdCnt (void)
- uint32 CFE_SB_RequestToSendEvent (uint32 TaskId, uint32 Bit)
- void CFE_SB_FinishSendEvent (uint32 TaskId, uint32 Bit)
- CFE_SB_DestinationD_t * CFE_SB_GetDestinationBlk (void)
- int32 CFE_SB_PutDestinationBlk (CFE_SB_DestinationD_t *Dest)
- int32 CFE_SB_AddDest (CFE_SB_RouteEntry_t *RouteEntry, CFE_SB_DestinationD_t *NewNode)
- int32 CFE_SB_RemoveDest (CFE_SB_RouteEntry_t *RouteEntry, CFE_SB_DestinationD_t *NodeToRemove)
- uint16 CFE_SB_MsgHdrSize (const CFE_SB_Msg_t *MsgPtr)

  Get the size of a software bus message header.
- int32 CFE_SB_NoopCmd (const CFE_SB_Noop_t *data)
- int32 CFE_SB_ResetCountersCmd (const CFE_SB_ResetCounters_t *data)
- int32 CFE_SB_EnableSubReportingCmd (const CFE_SB_EnableSubReporting_t *data)
- int32 CFE_SB_DisableSubReportingCmd (const CFE_SB_DisableSubReporting_t *data)
- int32 CFE_SB_SendHKTlmCmd (const CCSDS_CommandPacket_t *data)
- int32 CFE_SB_EnableRouteCmd (const CFE_SB_EnableRoute_t *data)
- int32 CFE_SB_DisableRouteCmd (const CFE_SB_DisableRoute_t *data)
- int32 CFE_SB_SendStatsCmd (const CFE_SB_SendSbStats_t *data)
- int32 CFE_SB_SendRoutingInfoCmd (const CFE_SB_SendRoutingInfo_t *data)
- int32 CFE_SB_SendPipeInfoCmd (const CFE_SB_SendPipeInfo_t *data)
- int32 CFE_SB_SendMapInfoCmd (const CFE_SB_SendMapInfo_t *data)
- int32 CFE_SB_SendPrevSubsCmd (const CFE_SB_SendPrevSubs_t *data)
- static bool CFE_SB_IsValidMsgId (CFE_SB_MsgId_t MsgId)

  Identifies whether a given CFE_SB_MsgId_t is valid.
- static bool CFE_SB_IsValidMsgKey (CFE_SB_MsgKey_t MsgKey)

  Identifies whether a given CFE_SB_MsgKey_t is valid.
- static bool CFE_SB_IsValidRouteIdx (CFE_SB_MsgRouteIdx_t RouteIdx)

  Identifies whether a given CFE_SB_MsgRouteIdx_t is valid.
- static CFE_SB_MsgKey_Atom_t CFE_SB_MsgKeyToValue (CFE_SB_MsgKey_t MsgKey)

  Converts between a CFE_SB_MsgKey_t and a raw value.
- static CFE_SB_MsgKey_t CFE_SB_ValueToMsgKey (CFE_SB_MsgKey_Atom_t KeyIdx)

  Converts between a CFE_SB_MsgKey_t and a raw value.
- static CFE_SB_MsgRouteIdx_t CFE_SB_ValueToRouteIdx (CFE_SB_MsgRouteIdx_Atom_t TableIdx)

  Converts between a CFE_SB_MsgRouteIdx_t and a raw value.
- static CFE_SB_MsgRouteIdx_Atom_t CFE_SB_RouteIdxToValue (CFE_SB_MsgRouteIdx_t RouteIdx)

  Converts between a CFE_SB_MsgRouteIdx_t and a raw value.

Variables

- cfe_sb_t CFE_SB

13.88.1 Macro Definition Documentation

13.88.1.1 CFE_SB_ACTIVE

```
#define CFE_SB_ACTIVE 1
```

Definition at line 61 of file cfe_sb_priv.h.

Referenced by CFE_SB_EnableRouteCmd(), and CFE_SB_SubscribeFull().

13.88.1.2 CFE_SB_CMD

```
#define CFE_SB_CMD 1
```

Definition at line 67 of file cfe_sb_priv.h.

13.88.1.3 CFE_SB_CMD_PIPE_DEPTH

```
#define CFE_SB_CMD_PIPE_DEPTH 32
```

Definition at line 85 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit().

13.88.1.4 CFE_SB_CMD_PIPE_NAME

```
#define CFE_SB_CMD_PIPE_NAME "SB_CMD_PIPE"
```

Definition at line 86 of file cfe_sb_priv.h.

Referenced by CFE_SB_AppInit().

13.88.1.5 CFE_SB_DENIED

```
#define CFE_SB_DENIED 0
```

Definition at line 78 of file cfe_sb_priv.h.

Referenced by CFE_SB_RequestToSendEvent().

13.88.1.6 CFE_SB_DISABLE

```
#define CFE_SB_DISABLE 0
```

Definition at line 75 of file cfe_sb_priv.h.

Referenced by CFE_SB_DisableSubReportingCmd(), and CFE_SB_EarlyInit().

13.88.1.7 CFE_SB_DO_NOT_INCREMENT

```
#define CFE_SB_DO_NOT_INCREMENT 0
```

Definition at line 81 of file cfe_sb_priv.h.

Referenced by CFE_SB_PassMsg(), and CFE_SB_ZeroCopyPass().

13.88.1.8 CFE_SB_DUPLICATE

```
#define CFE_SB_DUPLICATE 1
```

Definition at line 58 of file cfe_sb_priv.h.

Referenced by CFE_SB_DuplicateSubscribeCheck(), and CFE_SB_SubscribeFull().

13.88.1.9 CFE_SB_ENABLE

```
#define CFE_SB_ENABLE 1
```

Definition at line 76 of file cfe_sb_priv.h.

Referenced by CFE_SB_EnableSubReportingCmd(), and CFE_SB_SubscribeFull().

13.88.1.10 CFE_SB_FAILED

```
#define CFE_SB_FAILED 1
```

Definition at line 54 of file cfe_sb_priv.h.

Referenced by CFE_SB_ValidateMsgId(), and CFE_SB_ValidatePipeId().

13.88.1.11 CFE_SB_FILE_IO_ERR

```
#define CFE_SB_FILE_IO_ERR (-5)
```

Definition at line 95 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.88.1.12 CFE_SB_GET_BUF_ERR_EID_BIT

```
#define CFE_SB_GET_BUF_ERR_EID_BIT 1
```

Definition at line 99 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

13.88.1.13 CFE_SB_GLOBAL

```
#define CFE_SB_GLOBAL 0
```

Definition at line 63 of file cfe_sb_priv.h.

Referenced by CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_SendPrevSubsCmd(), CFE_SB_Subscribe(), CFE_SB_↩ SubscribeEx(), CFE_SB_SubscribeFull(), and CFE_SB_Unsubscribe().

13.88.1.14 CFE_SB_GRANTED

```
#define CFE_SB_GRANTED 1
```

Definition at line 79 of file cfe_sb_priv.h.

Referenced by CFE_SB_RequestToSendEvent(), and CFE_SB_SendMsgFull().

13.88.1.15 CFE_SB_IN_USE

```
#define CFE_SB_IN_USE 1
```

Definition at line 73 of file cfe_sb_priv.h.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), and CFE_SB_SendPipeInfo().

13.88.1.16 CFE_SB_INACTIVE

```
#define CFE_SB_INACTIVE 0
```

Definition at line 60 of file cfe_sb_priv.h.

Referenced by CFE_SB_DisableRouteCmd(), and CFE_SB_SendMsgFull().

13.88.1.17 CFE_SB_INCREMENT_TLM

```
#define CFE_SB_INCREMENT_TLM 1
```

Definition at line 82 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsg(), CFE_SB_SendMsgFull(), and CFE_SB_ZeroCopySend().

13.88.1.18 CFE_SB_INVALID_MSG_KEY

```
#define CFE_SB_INVALID_MSG_KEY ((CFE_SB_MsgKey_t){ .KeyIdx = 0 })
```

Definition at line 50 of file cfe_sb_priv.h.

13.88.1.19 CFE_SB_INVALID_PIPE

```
#define CFE_SB_INVALID_PIPE 0xFF
```

Definition at line 52 of file cfe_sb_priv.h.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetAvailPipeIdx(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeOpts(), CFE_SB_InitPipeTbl(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_↩ UnsubscribeFull().

13.88.1.20 CFE_SB_INVALID_ROUTE_IDX

```
#define CFE_SB_INVALID_ROUTE_IDX ((CFE_SB_MsgRouteIdx_t){ .RouteIdx = 0 })
```

Definition at line 49 of file cfe_sb_priv.h.

Referenced by CFE_SB_InitMsgMap(), and CFE_SB_RouteIdxPop_Unsync().

13.88.1.21 CFE_SB_LOCAL

```
#define CFE_SB_LOCAL 1
```

Definition at line 64 of file cfe_sb_priv.h.

Referenced by CFE_SB_SubscribeLocal(), CFE_SB_UnsubscribeLocal(), and CFE_SB_UnsubscribeWithAppId().

13.88.1.22 CFE_SB_MAIN_LOOP_ERR_DLY

```
#define CFE_SB_MAIN_LOOP_ERR_DLY 1000
```

Definition at line 84 of file cfe_sb_priv.h.

13.88.1.23 CFE_SB_MAX_CFG_FILE_EVENTS_TO_FILTER

```
#define CFE_SB_MAX_CFG_FILE_EVENTS_TO_FILTER 8
```

Definition at line 87 of file cfe_sb_priv.h.

13.88.1.24 CFE_SB_MAX_NUMBER_OF_MSG_KEYS

```
#define CFE_SB_MAX_NUMBER_OF_MSG_KEYS (1+CFE_PLATFORM_SB_HIGHEST_VALID_MSGID)
```

Definition at line 110 of file cfe_sb_priv.h.

Referenced by CFE_SB_InitMsgMap(), CFE_SB_IsValidMsgKey(), CFE_SB_SendMapInfo(), and CFE_SB_SendRtg↩Info().

13.88.1.25 CFE_SB_MSGID_LIM_ERR_EID_BIT

```
#define CFE_SB_MSGID_LIM_ERR_EID_BIT 2
```

Definition at line 100 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

13.88.1.26 CFE_SB_NO_DESTINATION

```
#define CFE_SB_NO_DESTINATION 0xFF
```

Definition at line 53 of file cfe_sb_priv.h.

13.88.1.27 CFE_SB_NO_DUPLICATE

```
#define CFE_SB_NO_DUPLICATE 0
```

Definition at line 57 of file cfe_sb_priv.h.

Referenced by CFE_SB_DuplicateSubscribeCheck().

13.88.1.28 CFE_SB_NOT_IN_USE

```
#define CFE_SB_NOT_IN_USE 0
```

Definition at line 72 of file cfe_sb_priv.h.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_GetAvailPipeIdx(), CFE_SB_InitPipeTbl(), and CFE_SB_↩ ValidatePipeId().

13.88.1.29 CFE_SB_PIPE_OVERFLOW

```
#define CFE_SB_PIPE_OVERFLOW (-1)
```

Definition at line 92 of file cfe_sb_priv.h.

13.88.1.30 CFE_SB_PIPE_WR_ERR

```
#define CFE_SB_PIPE_WR_ERR (-2)
```

Definition at line 93 of file cfe_sb_priv.h.

13.88.1.31 CFE_SB_Q_FULL_ERR_EID_BIT

```
#define CFE_SB_Q_FULL_ERR_EID_BIT 3
```

Definition at line 101 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

13.88.1.32 CFE_SB_Q_WR_ERR_EID_BIT

```
#define CFE_SB_Q_WR_ERR_EID_BIT 4
```

Definition at line 102 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

13.88.1.33 CFE_SB_QOS_LOW_PRIORITY

```
#define CFE_SB_QOS_LOW_PRIORITY 0
```

Definition at line 89 of file cfe_sb_priv.h.

Referenced by CFE_SB_EarlyInit().

13.88.1.34 CFE_SB_QOS_LOW_RELIABILITY

```
#define CFE_SB_QOS_LOW_RELIABILITY 0
```

Definition at line 90 of file cfe_sb_priv.h.

Referenced by CFE_SB_EarlyInit().

13.88.1.35 CFE_SB_SEND_NO_SUBS_EID_BIT

```
#define CFE_SB_SEND_NO_SUBS_EID_BIT 0
```

Definition at line 98 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

13.88.1.36 CFE_SB_SEND_ONECOPY

```
#define CFE_SB_SEND_ONECOPY 1
```

Definition at line 70 of file cfe_sb_priv.h.

Referenced by CFE_SB_PassMsg(), and CFE_SB_SendMsg().

13.88.1.37 CFE_SB_SEND_ZEROCOPY

```
#define CFE_SB_SEND_ZEROCOPY 0
```

Definition at line 69 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull(), CFE_SB_ZeroCopyPass(), and CFE_SB_ZeroCopySend().

13.88.1.38 CFE_SB_TLM

```
#define CFE_SB_TLM 0
```

Definition at line 66 of file cfe_sb_priv.h.

Referenced by CFE_SB_SendMsgFull().

13.88.1.39 CFE_SB_UNUSED_QUEUE

```
#define CFE_SB_UNUSED_QUEUE 0xFFFF
```

Definition at line 51 of file cfe_sb_priv.h.

Referenced by CFE_SB_DeletePipeFull(), and CFE_SB_InitPipeTbl().

13.88.1.40 CFE_SB_USECNT_ERR

```
#define CFE_SB_USECNT_ERR (-3)
```

Definition at line 94 of file cfe_sb_priv.h.

13.88.1.41 SB_DONT_CARE

```
#define SB_DONT_CARE 0
```

Definition at line 55 of file cfe_sb_priv.h.

13.88.2 Typedef Documentation

13.88.2.1 CFE_SB_MsgKey_Atom_t

```
typedef uint16 CFE_SB_MsgKey_Atom_t
```

Definition at line 125 of file cfe_sb_priv.h.

13.88.3 Function Documentation

13.88.3.1 CFE_SB_AddDest()

```
int32 CFE_SB_AddDest (
            CFE_SB_RouteEntry_t * RouteEntry,
            CFE_SB_DestinationD_t * NewNode )
```

Definition at line 762 of file cfe_sb_priv.c.

References CFE_SUCCESS, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, NULL, and CFE↩
_SB_DestinationD_t::Prev.

Referenced by CFE_SB_SubscribeFull().

13.88.3.2 CFE_SB_AppInit()

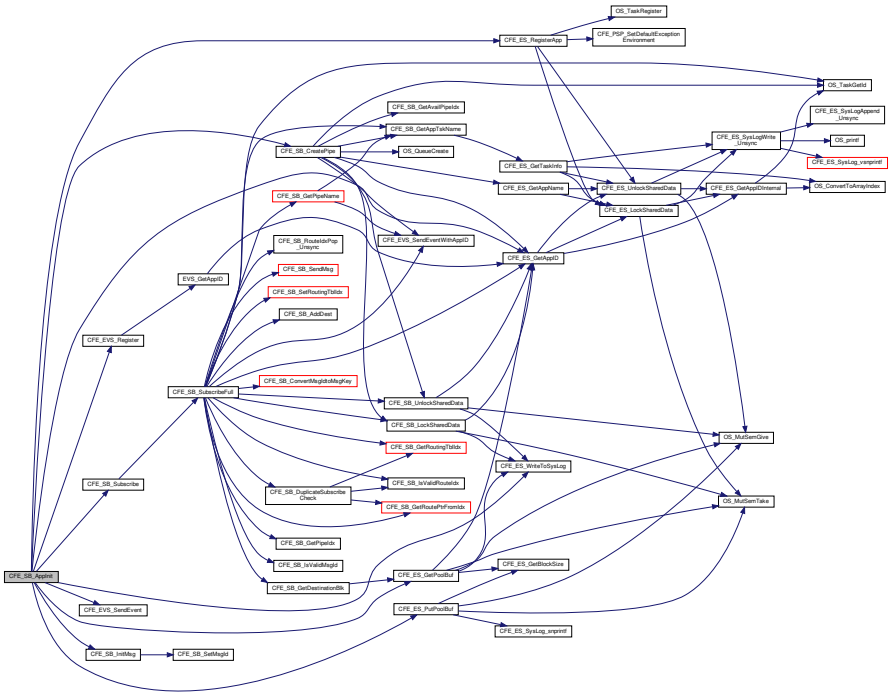
```
int32 CFE_SB_AppInit (
            void  )
```

Definition at line 136 of file cfe_sb_task.c.

References cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_ES_GetPoolBuf(), CFE_ES_PutPoolBuf(), CFE_ES_RegisterApp(), CFE_ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_PLATFORM_SB_BUF_MEMORY_BYTES, CFE_PLATFORM_SB_FILTER_MASK1, CFE_PLATFORM_SB_FILTER_MASK2, CFE_PLATFORM_SB_FILTER_MASK3, CFE_PLATFORM_SB_FILTER_MASK4, CFE_PLATFORM_SB_FILTER_MASK5, CFE_PLATFORM_SB_FILTER_MASK6, CFE_PLATFORM_SB_FILTER_MASK7, CFE_PLATFORM_SB_FILTER_MASK8, CFE_PLATFORM_SB_FILTERED_EVENT1, CFE_PLATFORM_SB_FILTERED_EVENT2, CFE_PLATFORM_SB_FILTERED_EVENT3, CFE_PLATFORM_SB_FILTERED_EVENT4, CFE_PLATFORM_SB_FILTERED_EVENT5, CFE_PLATFORM_SB_FILTERED_EVENT6, CFE_PLATFORM_SB_FILTERED_EVENT7, CFE_PLATFORM_SB_FILTERED_EVENT8, CFE_PLATFORM_SB_MAX_DEST_PER_PKT, CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_PLATFORM_SB_MAX_PIPE_DEPTH, CFE_PLATFORM_SB_MAX_PIPES, CFE_SB_ALLSUBS_TLM_MID, CFE_SB_CMD_MID, CFE_SB_CMD_PIPE_DEPTH, CFE_SB_CMD_PIPE_NAME, CFE_SB_CreatePipe(), CFE_SB_HK_TLM_MID, CFE_SB_INIT_EID, CFE_SB_InitMsg(), CFE_SB_ONESUB_TLM_MID, CFE_SB_SEND_HK_MID, CFE_SB_SET_MEMADDR, CFE_SB_Subscribe(), CFE_SUCCESS, cfe_sb_t::CmdPipe, cfe_sb_t::EventFilters, CFE_EVS_BinFilter_t::EventID, cfe_sb_t::HKTlmMsg, CFE_EVS_BinFilter_t::Mask, CFE_SB_StatsTlm_Payload_t::MaxMemAllowed, CFE_SB_StatsTlm_Payload_t::MaxMsgIdsAllowed, CFE_SB_StatsTlm_Payload_t::MaxPipeDepthAllowed, CFE_SB_StatsTlm_Payload_t::MaxPipesAllowed, CFE_SB_StatsTlm_Payload_t::MaxSubscriptionsAllowed, cfe_sb_t::Mem, CFE_SB_HousekeepingTlm_Payload_t::MemPoolHandle, NULL, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, cfe_sb_t::PrevSubMsg, cfe_sb_t::StatTlmMsg, and cfe_sb_t::SubRprtMsg.

Referenced by CFE_SB_TaskMain().

Here is the call graph for this function:



13.88.3.3 CFE_SB_ConvertMsgIdtoMsgKey()

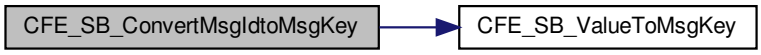
```
CFE_SB_MsgKey_t CFE_SB_ConvertMsgIdtoMsgKey (
            CFE_SB_MsgId_t MsgId )
```

Definition at line 111 of file cfe_sb_msg_id_util.c.

References CFE_SB_ValueToMsgKey().

Referenced by CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_RcvMsg(), CFE_SB_SendMsgFull(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.88.3.4 CFE_SB_DecrBufUseCnt()

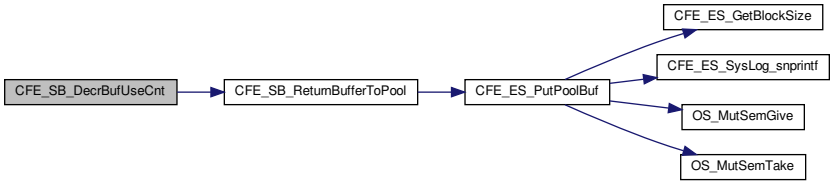
```
int32 CFE_SB_DecrBufUseCnt (
            CFE_SB_BufferD_t * bd )
```

Definition at line 178 of file cfe_sb_buf.c.

References CFE_SB_ReturnBufferToPool(), CFE_SUCCESS, and CFE_SB_BufferD_t::UseCount.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_RcvMsg(), and CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.88.3.5 CFE_SB_DeletePipeFull()

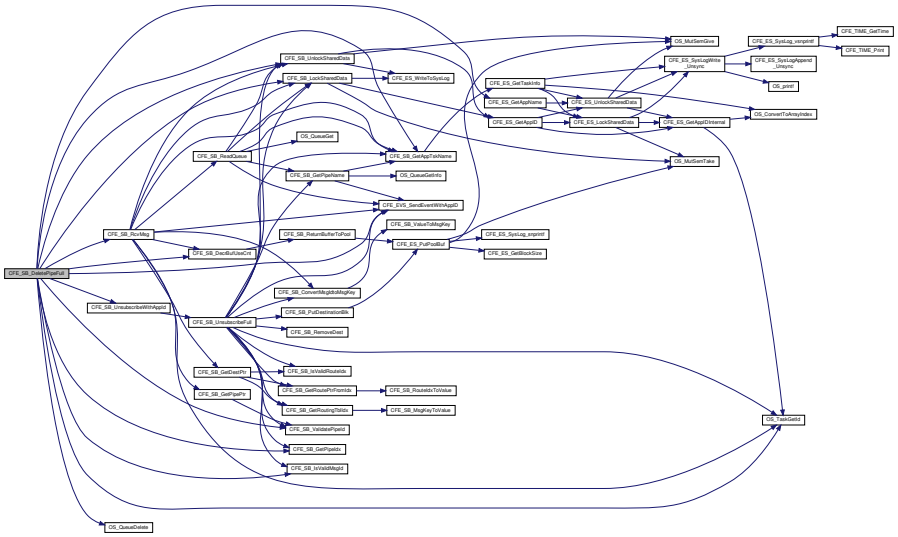
```
int32 CFE_SB_DeletePipeFull (
            CFE_SB_PipeId_t PipeId,
            uint32 AppId )
```

Definition at line 298 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppName(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_DecrBufUseCnt(), CFE_SB_DEL_PIPE_ERR1_EID, CFE_SB_DEL_PIPE_ERR2_EID, CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), CFE_SB_INVALID_PIPE, CFE_SB_IsValidMsgId(), CFE_SB_LockSharedData(), CFE_SB_NOT_IN_USE, CFE_SB_PIPE_DELETED_EID, CFE_SB_POLL, CFE_SB_RcvMsg(), CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_UnlockSharedData(), CFE_SB_UnsubscribeWithAppId(), CFE_SB_UNUSED_QUEUE, CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter, CFE_SB_PipeD_t::CurrentBuff, CFE_SB_PipeDepthStats_t::Depth, cfe_sb_t::HKTlmMsg, CFE_SB_PipeD_t::InUse, CFE_SB_PipeDepthStats_t::InUse, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, NULL, OS_MAX_API_NAME, OS_QueueDelete(), OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_PipeDepthStats_t::PeakInUse, CFE_SB_StatsTlm_Payload_t::PipeDepthStats, CFE_SB_PipeD_t::PipeId, CFE_SB_PipeDepthStats_t::PipeId, CFE_SB_StatsTlm_Payload_t::PipesInUse, cfe_sb_t::PipeTbl, cfe_sb_t::RoutingTbl, cfe_sb_t::StatTlmMsg, CFE_SB_PipeD_t::SysQueueId, and CFE_SB_PipeD_t::ToTrashBuff.

Referenced by CFE_SB_DeletePipe(), and CFE_SB_DeletePipeWithAppId().

Here is the call graph for this function:



13.88.3.6 CFE_SB_DeletePipeWithAppId()

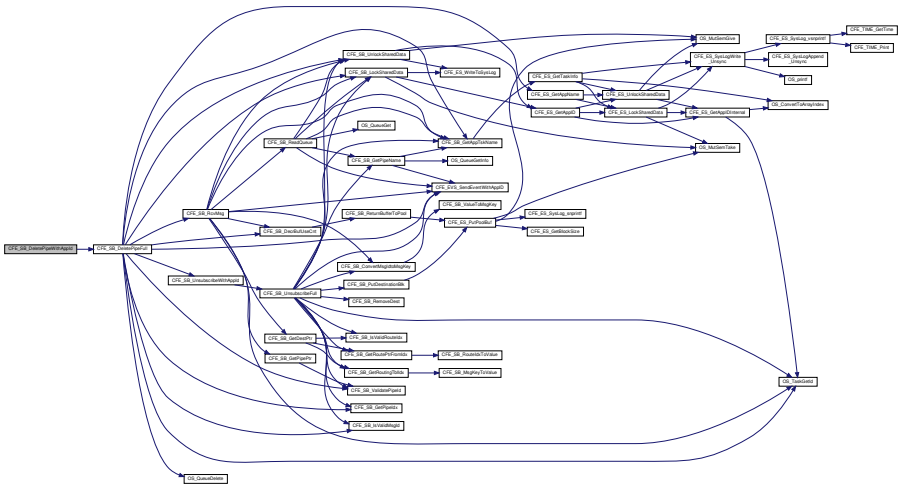
```
int32 CFE_SB_DeletePipeWithAppId (
            CFE_SB_PipeId_t PipeId,
            uint32 AppId )
```

Definition at line 271 of file cfe_sb_api.c.

References CFE_SB_DeletePipeFull().

Referenced by CFE_SB_CleanUpApp().

Here is the call graph for this function:



13.88.3.7 CFE_SB_DisableRouteCmd()

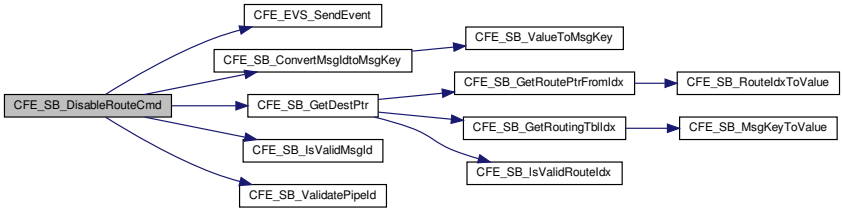
```
int32 CFE_SB_DisableRouteCmd (
            const CFE_SB_DisableRoute_t * data )
```

Definition at line 654 of file cfe_sb_task.c.

References CFE_SB_DestinationD_t::Active, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_DSBL_RTE1_EID, CFE_SB_DSBL_RTE2_EID, CFE_SB_DSBL_RTE3_EID, CFE_SB_GetDestPtr(), CFE_SB_INACTIVE, CFE_SB_IsValidMsgId(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, CFE_SB_RouteCmd_Payload_t::MsgId, NULL, CFE_SB_RouteCmd_t::Payload, CFE_SB_HousekeepingTlm_t::Payload, and CFE_SB_RouteCmd_Payload_t::Pipe.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.8 CFE_SB_DisableSubReportingCmd()

```
int32 CFE_SB_DisableSubReportingCmd (
            const CFE_SB_DisableSubReporting_t * data )
```

Definition at line 513 of file cfe_sb_task.c.

References CFE_SB_DISABLE, CFE_SB_SetSubscriptionReporting(), and CFE_SUCCESS.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:

13.88.3.9 CFE_SB_DuplicateSubscribeCheck()

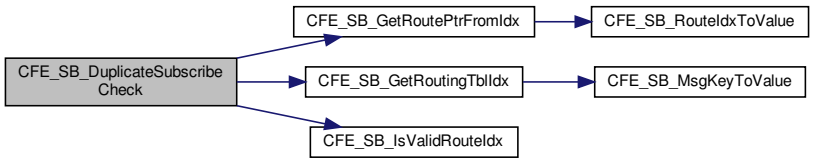
```
int32 CFE_SB_DuplicateSubscribeCheck (
            CFE_SB_MsgKey_t MsgKey,
            CFE_SB_PipeId_t PipeId )
```

Definition at line 505 of file cfe_sb_priv.c.

References CFE_SB_DUPLICATE, CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_IsValid↩RouteIdx(), CFE_SB_NO_DUPLICATE, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, and N↩ULL.

Referenced by CFE_SB_SubscribeFull().

Here is the call graph for this function:



13.88.3.10 CFE_SB_EnableRouteCmd()

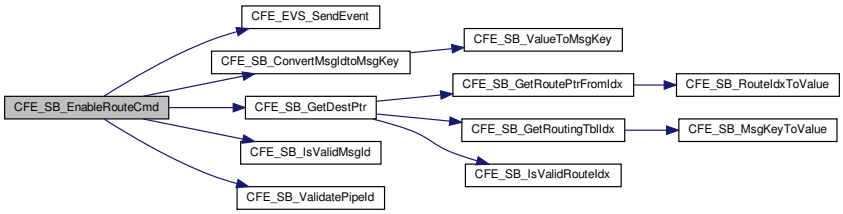
```
int32 CFE_SB_EnableRouteCmd (
            const CFE_SB_EnableRoute_t * data )
```

Definition at line 592 of file cfe_sb_task.c.

References CFE_SB_DestinationD_t::Active, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE↩ _EVS_SendEvent(), CFE_SB_ACTIVE, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_ENBL_RTE1_EID, CFE_SB_↩ ENBL_RTE2_EID, CFE_SB_ENBL_RTE3_EID, CFE_SB_GetDestPtr(), CFE_SB_IsValidMsgId(), CFE_SB_Validate↩ PipeId(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_HousekeepingTlm↩ _Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, CFE_SB_RouteCmd_Payload_t::MsgId, NULL, CFE_SB↩ _RouteCmd_t::Payload, CFE_SB_HousekeepingTlm_t::Payload, and CFE_SB_RouteCmd_Payload_t::Pipe.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.11 CFE_SB_EnableSubReportingCmd()

```
int32 CFE_SB_EnableSubReportingCmd (
            const CFE_SB_EnableSubReporting_t * data )
```

Definition at line 500 of file cfe_sb_task.c.

References CFE_SB_ENABLE, CFE_SB_SetSubscriptionReporting(), and CFE_SUCCESS.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:

13.88.3.12 CFE_SB_FileWriteByteCntErr()

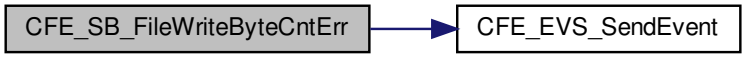
```
void CFE_SB_FileWriteByteCntErr (
            const char * Filename,
            uint32 Requested,
            uint32 Actual )
```

Definition at line 1277 of file cfe_sb_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), and CFE_SB_FILEWRITE_ERR_EID.

Referenced by CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

Here is the call graph for this function:



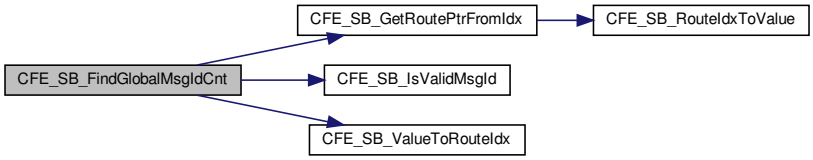
13.88.3.13 CFE_SB_FindGlobalMsgIdCnt()

```
uint32 CFE_SB_FindGlobalMsgIdCnt (
            void  )
```

Definition at line 1198 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GLOBAL, CFE_SB_IsValidMsgId(), CFE_SB_ValueToRouteIdx(), CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, NULL, and CFE_SB_DestinationD_t::Scope.

Here is the call graph for this function:



13.88.3.14 CFE_SB_FinishSendEvent()

```
void CFE_SB_FinishSendEvent (
            uint32 TaskId,
            uint32 Bit )
```

Definition at line 739 of file cfe_sb_priv.c.

References CFE_CLR, CFE_SB, OS_ConvertToArrayIndex(), and cfe_sb_t::StopRecurseFlags.

Referenced by CFE_SB_SendMsgFull().

Here is the call graph for this function:

13.88.3.15 CFE_SB_GetAppTskName()

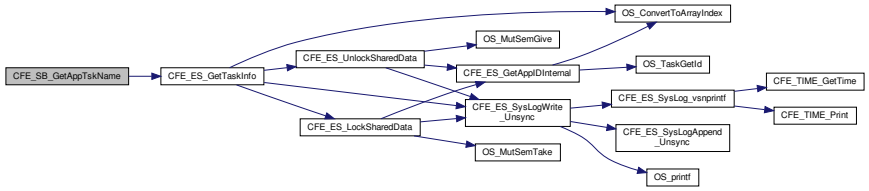
```
char* CFE_SB_GetAppTskName (
            uint32 TaskId,
            char * FullName )
```

Definition at line 627 of file cfe_sb_priv.c.

References CFE_ES_TaskInfo_t::AppName, CFE_ES_GetTaskInfo(), CFE_SUCCESS, OS_MAX_API_NAME, and CFE_ES_TaskInfo_t::TaskName.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.88.3.16 CFE_SB_GetAvailPipeIdx()

```
CFE_SB_PipeId_t CFE_SB_GetAvailPipeIdx (
            void  )
```

Definition at line 161 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_INVALID_PIPE, CFE_SB_NOT_IN_USE, CFE_SB_PipeD_t::InUse, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_CreatePipe().

13.88.3.17 CFE_SB_GetBufferFromCaller()

```
CFE_SB_BufferD_t* CFE_SB_GetBufferFromCaller (
            CFE_SB_MsgId_t MsgId,
            void * Address )
```

Definition at line 117 of file cfe_sb_buf.c.

References CFE_SB_BufferD_t::MsgId.

Referenced by CFE_SB_SendMsgFull().

13.88.3.18 CFE_SB_GetBufferFromPool()

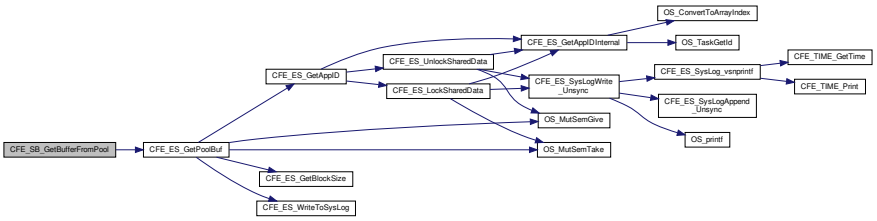
```
CFE_SB_BufferD_t* CFE_SB_GetBufferFromPool (
            CFE_SB_MsgId_t MsgId,
            uint16 Size )
```

Definition at line 60 of file cfe_sb_buf.c.

References CFE_ES_GetPoolBuf(), CFE_SB, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, CF↩E_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, CFE_SB_StatsTlm_Payload_t::PeakSB↩BuffersInUse, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, and cfe_sb_t::Stat↩TlmMsg.

Referenced by CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.88.3.19 CFE_SB_GetDestinationBlk()

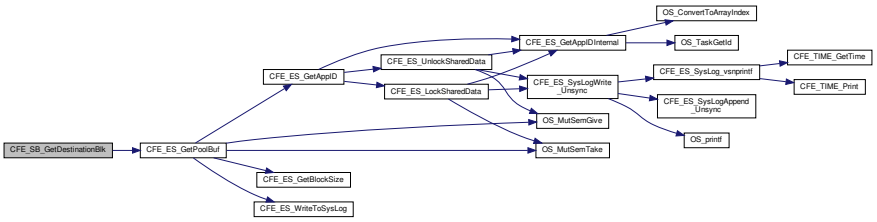
```
CFE_SB_DestinationD_t* CFE_SB_GetDestinationBlk (
            void  )
```

Definition at line 209 of file cfe_sb_buf.c.

References CFE_ES_GetPoolBuf(), CFE_SB, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, C↩FE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, CFE_SB_MemParams_t::PoolHdl, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_SubscribeFull().

Here is the call graph for this function:



13.88.3.20 CFE_SB_GetDestPtr()

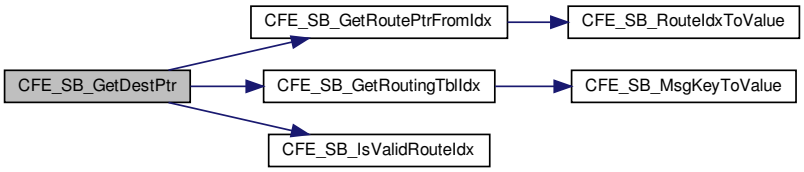
```
CFE_SB_DestinationD_t* CFE_SB_GetDestPtr (
            CFE_SB_MsgKey_t MsgKey,
            CFE_SB_PipeId_t PipeId )
```

Definition at line 384 of file cfe_sb_priv.c.

References CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_IsValidRouteIdx(), CFE_SB_↩ RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, and NULL.

Referenced by CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), and CFE_SB_RcvMsg().

Here is the call graph for this function:



13.88.3.21 CFE_SB_GetPipeIdx()

```
uint8 CFE_SB_GetPipeIdx (
            CFE_SB_PipeId_t PipeId )
```

Definition at line 250 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_INVALID_PIPE, CFE_SB_PipeD_t::InUse, CF↩ E_SB_PipeD_t::PipeId, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_GetPipeOpts(), CFE_SB_SetPipeOpts(), CFE_SB_Subscribe↩ Full(), and CFE_SB_UnsubscribeFull().

13.88.3.22 CFE_SB_GetPipePtr()

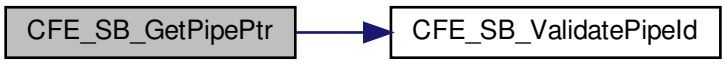
```
CFE_SB_PipeD_t* CFE_SB_GetPipePtr (
            CFE_SB_PipeId_t PipeId )
```

Definition at line 352 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_ValidatePipeId(), CFE_SUCCESS, NULL, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_RcvMsg(), and CFE_SB_SendRtgInfo().

Here is the call graph for this function:



13.88.3.23 CFE_SB_GetPktType()

```
uint8 CFE_SB_GetPktType (
            CFE_SB_MsgId_t MsgId )
```

Definition at line 675 of file cfe_sb_priv.c.

References CFE_SB_RD_TYPE_FROM_MSGID, and CFE_TST.

Referenced by CFE_SB_SendMsgFull().

13.88.3.24 CFE_SB_GetRoutePtrFromIdx()

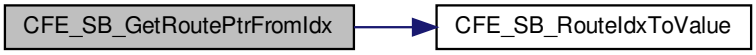
```
CFE_SB_RouteEntry_t* CFE_SB_GetRoutePtrFromIdx (
            CFE_SB_MsgRouteIdx_t RouteIdx )
```

Definition at line 486 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_RouteIdxToValue(), and cfe_sb_t::RoutingTbl.

Referenced by CFE_SB_DuplicateSubscribeCheck(), CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_GetDestPtr(), CFE↩
_SB_SendMapInfo(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRtgInfo(), CFE_SB_↩
SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.88.3.25 CFE_SB_GetRoutingTblIdx()

```
CFE_SB_MsgRouteIdx_t CFE_SB_GetRoutingTblIdx (
            CFE_SB_MsgKey_t MsgKey )
```

Definition at line 433 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_MsgKeyToValue(), and cfe_sb_t::MsgMap.

Referenced by CFE_SB_DuplicateSubscribeCheck(), CFE_SB_GetDestPtr(), CFE_SB_SendMapInfo(), CFE_SB_↩ SendMsgFull(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

Here is the call graph for this function:

13.88.3.26 CFE_SB_IncrCmdCtr()

```
void CFE_SB_IncrCmdCtr (
            int32 status )
```

Definition at line 1253 of file cfe_sb_task.c.

References CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_Housekeeping↩ Tlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

13.88.3.27 CFE_SB_InitBuffers()

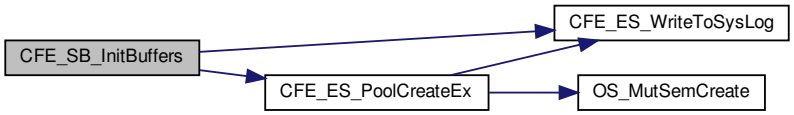
```
int32 CFE_SB_InitBuffers (
            void  )
```

Definition at line 152 of file cfe_sb_init.c.

References CFE_ES_MAX_MEMPOOL_BLOCK_SIZES, CFE_ES_NO_MUTEX, CFE_ES_PoolCreateEx(), CFE_ES_WriteToSysLog(), CFE_PLATFORM_SB_BUF_MEMORY_BYTES, CFE_SB, CFE_SB_MemPoolDefSize, CFE_SUCCESS, cfe_sb_t::Mem, and CFE_SB_MemParams_t::PoolHdl.

Referenced by CFE_SB_EarlyInit().

Here is the call graph for this function:



13.88.3.28 CFE_SB_InitIdxStack()

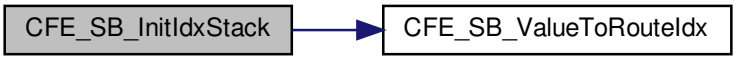
```
void CFE_SB_InitIdxStack (
            void  )
```

Definition at line 104 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_ValueToRouteIdx(), cfe_sb_t::RouteIdxStack, and cfe_sb_t::RouteIdxTop.

Referenced by CFE_SB_EarlyInit().

Here is the call graph for this function:



13.88.3.29 CFE_SB_InitMsgMap()

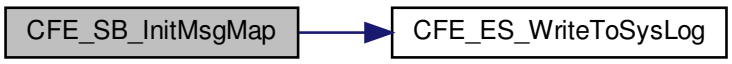
```
void CFE_SB_InitMsgMap (
            void  )
```

Definition at line 217 of file cfe_sb_init.c.

References CFE_ES_WriteToSysLog(), CFE_SB, CFE_SB_INVALID_ROUTE_IDX, CFE_SB_MAX_NUMBER_OF_↩ MSG_KEYS, and cfe_sb_t::MsgMap.

Referenced by CFE_SB_EarlyInit().

Here is the call graph for this function:



13.88.3.30 CFE_SB_InitPipeTbl()

```
void CFE_SB_InitPipeTbl (
            void  )
```

Definition at line 188 of file cfe_sb_init.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_INVALID_PIPE, CFE_SB_NOT_IN_USE, CFE↩ _SB_UNUSED_QUEUE, CFE_SB_PipeD_t::CurrentBuff, CFE_SB_PipeD_t::InUse, NULL, CFE_SB_PipeD_t::PipeId, cfe_sb_t::PipeTbl, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_EarlyInit().

13.88.3.31 CFE_SB_InitRoutingTbl()

```
void CFE_SB_InitRoutingTbl (
            void  )
```

Definition at line 252 of file cfe_sb_init.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_INVALID_MSG_ID, CFE_SB_RouteEntry_↩ t::Destinations, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, NULL, cfe_sb_t::RoutingTbl, and CFE_SB_RouteEntry_t::SeqCnt.

Referenced by CFE_SB_EarlyInit().

13.88.3.32 CFE_SB_IsValidMsgId()

```
static bool CFE_SB_IsValidMsgId (
            CFE_SB_MsgId_t MsgId )  [inline], [static]
```

Implements a basic sanity check on the value provided

Returns

if sanity checks passed, false otherwise.

Definition at line 482 of file cfe_sb_priv.h.

References CFE_PLATFORM_SB_HIGHEST_VALID_MSGID, and CFE_SB_INVALID_MSG_ID.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_SB_↩ FindGlobalMsgIdCnt(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SubscribeFull(), CFE_S↩ B_UnsubscribeFull(), and CFE_SB_ValidateMsgId().

13.88.3.33 CFE_SB_IsValidMsgKey()

```
static bool CFE_SB_IsValidMsgKey (
            CFE_SB_MsgKey_t MsgKey )  [inline], [static]
```

Implements a basic sanity check on the value provided

Returns

if sanity checks passed, false otherwise.

Definition at line 495 of file cfe_sb_priv.h.

References CFE_SB_MAX_NUMBER_OF_MSG_KEYS, and CFE_SB_MsgKey_t::KeyIdx.

13.88.3.34 CFE_SB_IsValidRouteIdx()

```
static bool CFE_SB_IsValidRouteIdx (
            CFE_SB_MsgRouteIdx_t RouteIdx )  [inline], [static]
```

Implements a basic sanity check on the value provided

Returns

if sanity checks passed, false otherwise.

Definition at line 507 of file cfe_sb_priv.h.

References CFE_PLATFORM_SB_MAX_MSG_IDS, and CFE_SB_MsgRouteIdx_t::RouteIdx.

Referenced by CFE_SB_DuplicateSubscribeCheck(), CFE_SB_GetDestPtr(), CFE_SB_SendMapInfo(), CFE_SB_↩ SendMsgFull(), CFE_SB_SendRtgInfo(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

13.88.3.35 CFE_SB_LockSharedData()

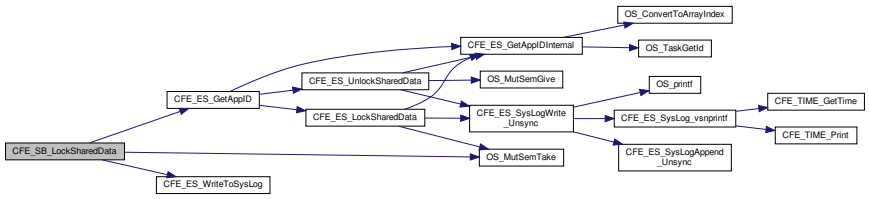
```
void CFE_SB_LockSharedData (
            const char * FuncName,
            int32 LineNumber )
```

Definition at line 282 of file cfe_sb_priv.c.

References CFE_ES_GetAppID(), CFE_ES_WriteToSysLog(), CFE_SB, OS_MutSemTake(), OS_SUCCESS, and cfe_sb_t::SharedDataMutexId.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipe↩ IdByName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE↩ _SB_SendPrevSubsCmd(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_S↩ B_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

Here is the call graph for this function:



13.88.3.36 CFE_SB_MsgHdrSize()

```
uint16 CFE_SB_MsgHdrSize (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the number of bytes in a software bus message header. This can be used for sizing buffers that need to store SB messages. SB message header formats can be different for each deployment of the cFE. So, applications should use this function and avoid hard coding their buffer sizes.

Assumptions, External Events, and Notes:

- For statically defined messages, a function call will not work. The macros CFE_SB_CMD_HDR_SIZE and CFE_SB_TLM_HDR_SIZE are available for use in static message buffer sizing or structure definitions.

Parameters

in	*MsgPtr	The message ID to calculate header size for. The size of the message header may depend on the MsgId in some implementations. For example, if SB messages are implemented as CCSDS packets, the size of the header is different for command vs. telemetry packets.

The number of bytes in the software bus message header for messages with the given `MsgId`. endstmt

Returns

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum

Definition at line 110 of file cfe_sb_util.c.

References CCSDS_CMD, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CFE_SB_CMD_HDR_SIZE, and CFE_SB_TLM_HDR_SIZE.

Referenced by CFE_SB_GetUserData(), CFE_SB_GetUserDataLength(), and CFE_SB_SetUserDataLength().

13.88.3.37 CFE_SB_MsgKeyToValue()

```
static CFE_SB_MsgKey_Atom_t CFE_SB_MsgKeyToValue (
            CFE_SB_MsgKey_t MsgKey )  [inline], [static]
```

Converts the supplied value into a "bare number" suitable for performing array lookups or other tasks for which the holding structure cannot be used directly.

Use with caution, as this removes the type safety information from the value.

Note

It is assumed the value has already been validated using CFE_SB_IsValidMsgKey()

Returns

underlying index value

Definition at line 524 of file cfe_sb_priv.h.

References CFE_SB_MsgKey_t::KeyIdx.

Referenced by CFE_SB_GetRoutingTblIdx(), and CFE_SB_SetRoutingTblIdx().

13.88.3.38 CFE_SB_NoopCmd()

```
int32 CFE_SB_NoopCmd (
            const CFE_SB_Noop_t * data )
```

Definition at line 466 of file cfe_sb_task.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MAJOR_VERSION, CFE_MIN↩ OR_VERSION, CFE_MISSION_REV, CFE_REVISION, CFE_SB_CMD0_RCVD_EID, CFE_SUCCESS, CFE_SB_↩ HousekeepingTlm_Payload_t::CommandCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:

13.88.3.39 CFE_SB_ProcessCmdPipePkt()

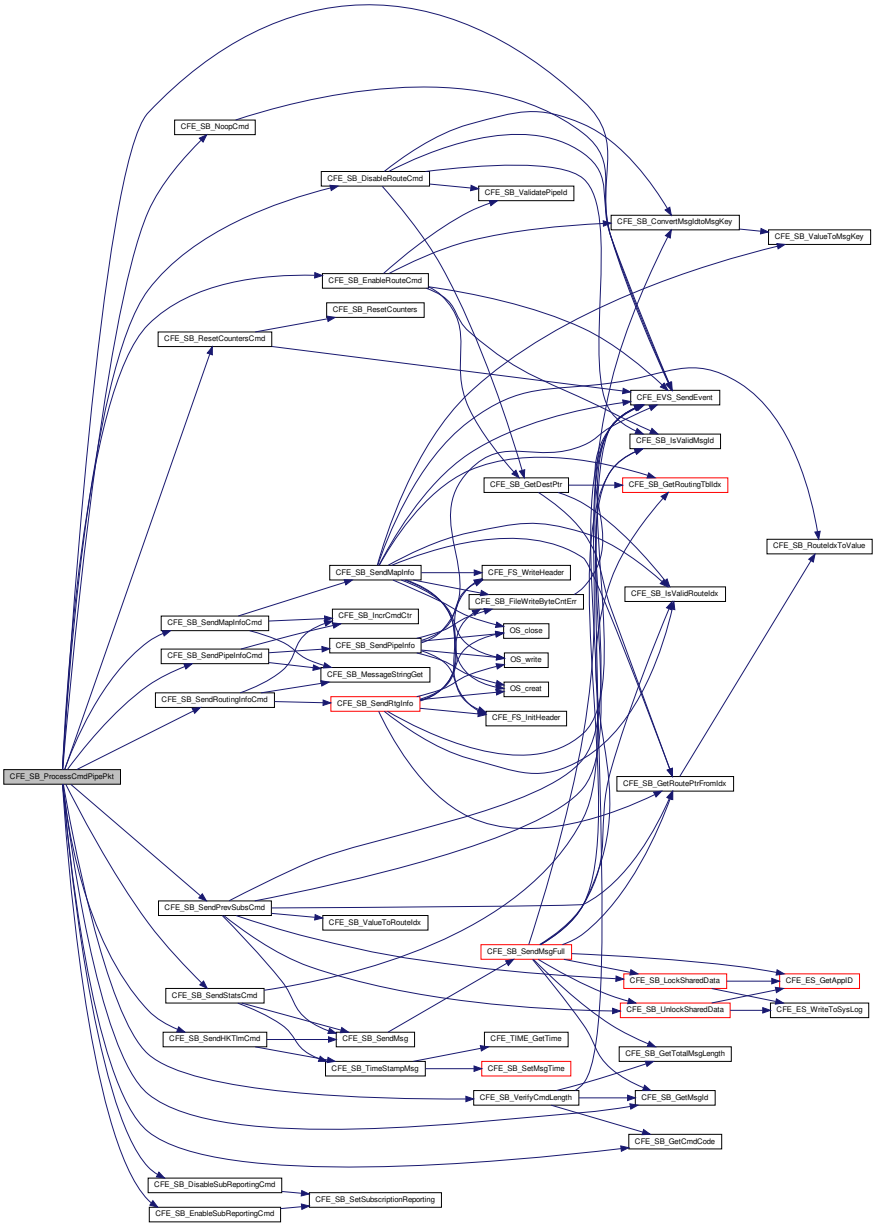
```
void CFE_SB_ProcessCmdPipePkt (
            void  )
```

Definition at line 349 of file cfe_sb_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_BAD_CMD_CODE_EID, CFE_SB_↩ BAD_MSGID_EID, CFE_SB_CMD_MID, CFE_SB_DISABLE_ROUTE_CC, CFE_SB_DISABLE_SUB_REPORTING↩ _CC, CFE_SB_DisableRouteCmd(), CFE_SB_DisableSubReportingCmd(), CFE_SB_ENABLE_ROUTE_CC, CFE_↩ SB_ENABLE_SUB_REPORTING_CC, CFE_SB_EnableRouteCmd(), CFE_SB_EnableSubReportingCmd(), CFE_S↩ B_GetCmdCode(), CFE_SB_GetMsgId(), CFE_SB_NOOP_CC, CFE_SB_NoopCmd(), CFE_SB_RESET_COUNTE↩ RS_CC, CFE_SB_ResetCountersCmd(), CFE_SB_SEND_HK_MID, CFE_SB_SEND_MAP_INFO_CC, CFE_SB_SE↩ ND_PIPE_INFO_CC, CFE_SB_SEND_PREV_SUBS_CC, CFE_SB_SEND_ROUTING_INFO_CC, CFE_SB_SEND↩ _SB_STATS_CC, CFE_SB_SendHKTlmCmd(), CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), CFE_↩ SB_SendPrevSubsCmd(), CFE_SB_SendRoutingInfoCmd(), CFE_SB_SendStatsCmd(), CFE_SB_VerifyCmdLength(), cfe_sb_t::CmdPipePktPtr, CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_TaskMain().

Here is the call graph for this function:



13.88.3.40 CFE_SB_PutDestinationBlk()

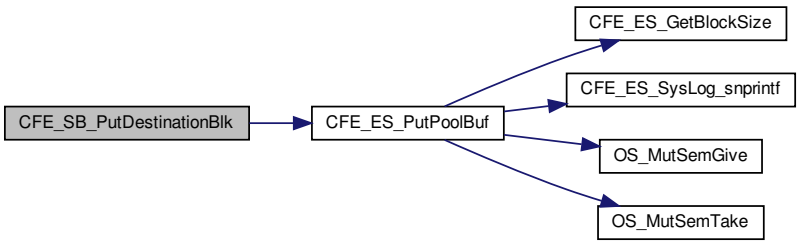
```
int32 CFE_SB_PutDestinationBlk (
            CFE_SB_DestinationD_t * Dest )
```

Definition at line 244 of file cfe_sb_buf.c.

References CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, NULL, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_UnsubscribeFull().

Here is the call graph for this function:



13.88.3.41 CFE_SB_ReadQueue()

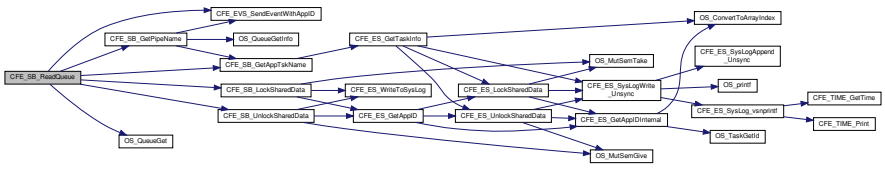
```
int32 CFE_SB_ReadQueue (
            CFE_SB_PipeD_t * PipeDscPtr,
            uint32 TskId,
            CFE_SB_TimeOut_t Time_Out,
            CFE_SB_BufferD_t ** Message )
```

Definition at line 2270 of file cfe_sb_api.c.

References cfe_sb_t::AppId, CFE_EVS_EventType_ERROR, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_GetAppTskName(), CFE_SB_GetPipeName(), CFE_SB_LockSharedData(), CFE_SB_NO_MESSAGE, CFE_SB_PEND_FOREVER, CFE_SB_PIPE_RD_ERR, CFE_SB_POLL, CFE_SB_Q_RD_ERR_EID, CFE_SB_TIME_OUT, CFE_SB_UnlockSharedData(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::InternalErrorCounter, OS_CHECK, OS_MAX_API_NAME, OS_PEND, OS_QUEUE_EMPTY, OS_QUEUE_TIMEOUT, OS_QueueGet(), OS_SUCCESS, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_PipeD_t::PipeId, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_RcvMsg().

Here is the call graph for this function:



13.88.3.42 CFE_SB_ReleaseBuffer()

```
void CFE_SB_ReleaseBuffer (
            CFE_SB_BufferD_t * bd,
            CFE_SB_DestinationD_t * dest )
```

13.88.3.43 CFE_SB_RemoveDest()

```
int32 CFE_SB_RemoveDest (
            CFE_SB_RouteEntry_t * RouteEntry,
            CFE_SB_DestinationD_t * NodeToRemove )
```

Definition at line 810 of file cfe_sb_priv.c.

References CFE_SUCCESS, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, NULL, and CFE↩_SB_DestinationD_t::Prev.

Referenced by CFE_SB_UnsubscribeFull().

13.88.3.44 CFE_SB_RequestToSendEvent()

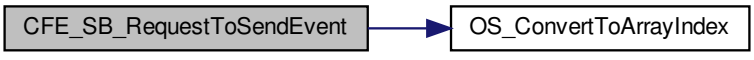
```
uint32 CFE_SB_RequestToSendEvent (
            uint32 TaskId,
            uint32 Bit )
```

Definition at line 707 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_DENIED, CFE_SB_GRANTED, CFE_SET, CFE_TST, OS_ConvertToArrayIndex(), and cfe_sb_t::StopRecurseFlags.

Referenced by CFE_SB_SendMsgFull().

Here is the call graph for this function:



13.88.3.45 CFE_SB_ResetCounters()

```
void CFE_SB_ResetCounters (
            void  )
```

Definition at line 562 of file cfe_sb_task.c.

References CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_HousekeepingTlm_Payload_t::↩ CommandErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter, CFE_SB_Housekeeping↩ Tlm_Payload_t::DuplicateSubscriptionsCounter, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::↩ InternalErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::MsgLimitErrorCounter, CFE_SB_Housekeeping↩ Tlm_Payload_t::MsgReceiveErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::MsgSendErrorCounter, CFE↩ _SB_HousekeepingTlm_Payload_t::NoSubscribersCounter, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_↩ HousekeepingTlm_Payload_t::PipeOverflowErrorCounter, and CFE_SB_HousekeepingTlm_Payload_t::Subscribe↩ ErrorCounter.

Referenced by CFE_SB_ResetCountersCmd().

13.88.3.46 CFE_SB_ResetCountersCmd()

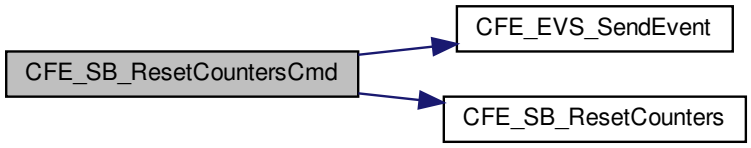
```
int32 CFE_SB_ResetCountersCmd (
            const CFE_SB_ResetCounters_t * data )
```

Definition at line 483 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_SB_CMD1_RCVD_EID, CFE_SB_Reset↩ Counters(), and CFE_SUCCESS.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.47 CFE_SB_ResetCounts()

```
void CFE_SB_ResetCounts (
            void  )
```

13.88.3.48 CFE_SB_ReturnBufferToPool()

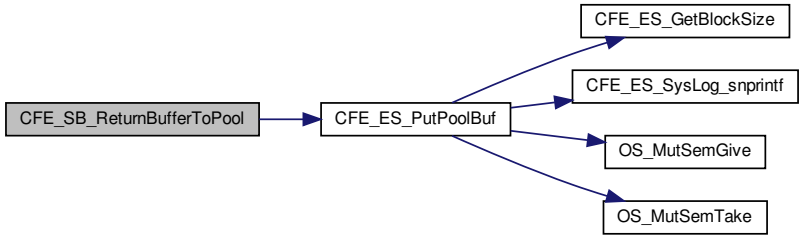
```
int32 CFE_SB_ReturnBufferToPool (
            CFE_SB_BufferD_t * bd )
```

Definition at line 143 of file cfe_sb_buf.c.

References CFE_ES_PutPoolBuf(), CFE_SB, CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::MemInUse, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, CFE_SB_StatsTlm_Payload_t::SBBuffersInUse, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_DecrBufUseCnt().

Here is the call graph for this function:



13.88.3.49 CFE_SB_RouteIdxPop_Unsync()

```
CFE_SB_MsgRouteIdx_t CFE_SB_RouteIdxPop_Unsync (
            void  )
```

Definition at line 195 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_INVALID_ROUTE_IDX, cfe_sb_t::RouteIdxStack, and cfe_sb_t::RouteIdxTop.

Referenced by CFE_SB_SubscribeFull().

13.88.3.50 CFE_SB_RouteIdxPush_Unsync()

```
void CFE_SB_RouteIdxPush_Unsync (
            CFE_SB_MsgRouteIdx_t idx )
```

Definition at line 228 of file cfe_sb_priv.c.

References CFE_SB, cfe_sb_t::RouteIdxStack, and cfe_sb_t::RouteIdxTop.

13.88.3.51 CFE_SB_RouteIdxToValue()

```
static CFE_SB_MsgRouteIdx_Atom_t CFE_SB_RouteIdxToValue (
            CFE_SB_MsgRouteIdx_t RouteIdx )  [inline], [static]
```

Converts the supplied value into a "bare number" suitable for performing array lookups or other tasks for which the holding structure cannot be used directly.

Use with caution, as this removes the type safety information from the value.

Note

It is assumed the value has already been validated using CFE_SB_IsValidRouteIdx()

Returns

underlying index value

Definition at line 565 of file cfe_sb_priv.h.

References CFE_SB_MsgRouteIdx_t::RouteIdx.

Referenced by CFE_SB_GetRoutePtrFromIdx(), and CFE_SB_SendMapInfo().

13.88.3.52 CFE_SB_SendHKTlmCmd()

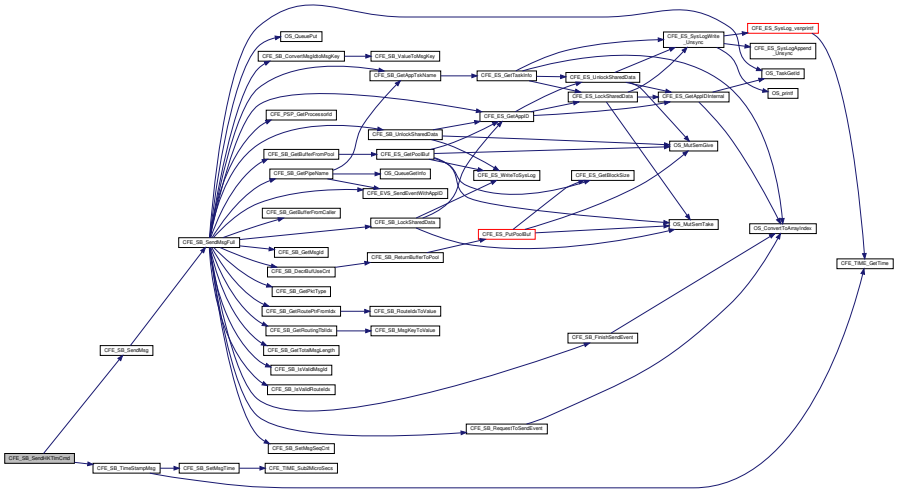
```
int32 CFE_SB_SendHKTlmCmd (
            const CCSDS_CommandPacket_t * data )
```

Definition at line 535 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_BUF_MEMORY_BYTES, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), CFE_SUCCESS, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::MemInUse, CFE_SB_StatsTlm_Payload_t::MemInUse, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMemInUse, cfe_sb_t::StatTlmMsg, and CFE_SB_HousekeepingTlm_Payload_t::UnmarkedMem.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.53 CFE_SB_SendMapInfo()

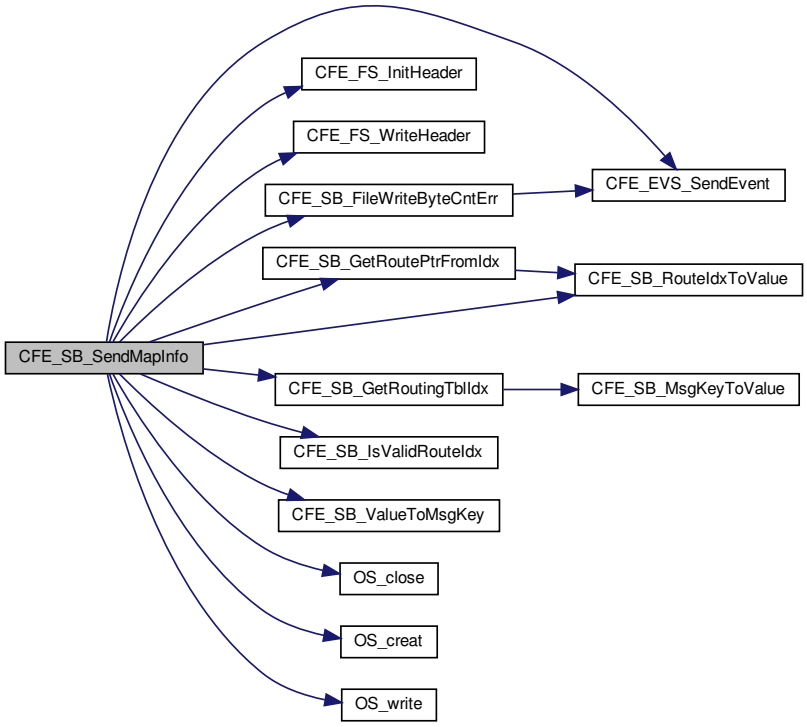
```
int32 CFE_SB_SendMapInfo (
            const char * Filename )
```

Definition at line 1017 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_Init↩
Header(), CFE_FS_SubType_SB_MAPDATA, CFE_FS_WriteHeader(), CFE_SB_FILE_IO_ERR, CFE_SB_FileWrite↩
ByteCntErr(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_IsValidRouteIdx(), CFE_SB_↩
MAX_NUMBER_OF_MSG_KEYS, CFE_SB_RouteIdxToValue(), CFE_SB_SND_RTG_EID, CFE_SB_SND_RTG_E↩
RR1_EID, CFE_SB_ValueToMsgKey(), CFE_SUCCESS, CFE_SB_MsgMapFileEntry_t::Index, CFE_SB_RouteEntry↩
_t::MsgId, CFE_SB_MsgMapFileEntry_t::MsgId, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_write(), and OS_↩
WRITE_ONLY.

Referenced by CFE_SB_SendMapInfoCmd().

Here is the call graph for this function:



13.88.3.54 CFE_SB_SendMapInfoCmd()

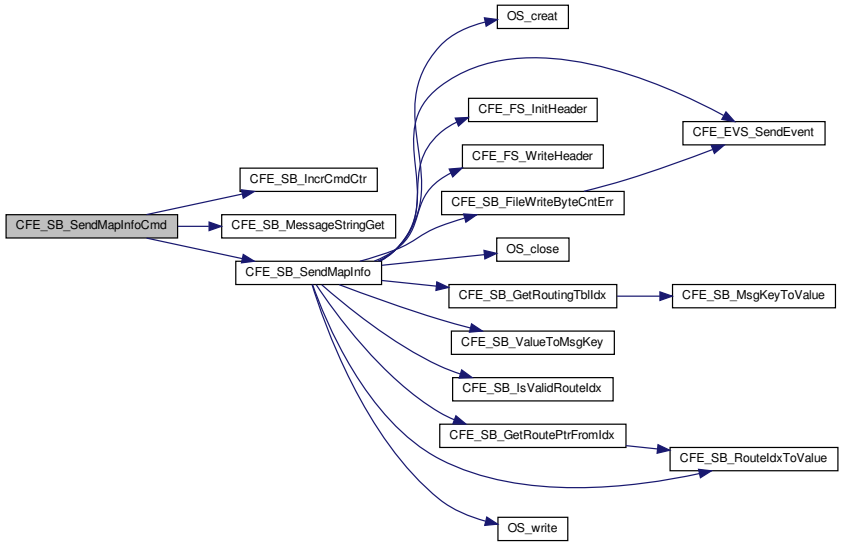
```
int32 CFE_SB_SendMapInfoCmd (
            const CFE_SB_SendMapInfo_t * data )
```

Definition at line 800 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME, CFE_SB_IncrCmdCtr(), CFE_SB_MessageString↩Get(), CFE_SB_SendMapInfo(), CFE_SUCCESS, CFE_SB_WriteFileInfoCmd_Payload_t::Filename, OS_MAX_PAT↩H_LEN, and CFE_SB_WriteFileInfoCmd_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.55 CFE_SB_SendMsgFull()

```
int32 CFE_SB_SendMsgFull (
            CFE_SB_Msg_t * MsgPtr,
            uint32 TlmCntIncrements,
            uint32 CopyMode )
```

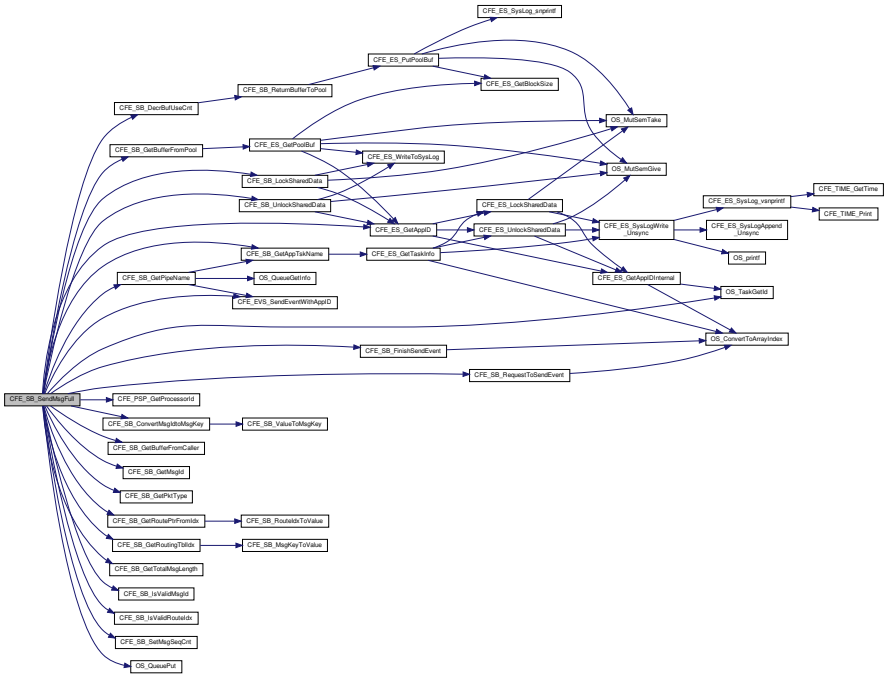
Definition at line 1390 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_SB_SenderId_t::AppName, CFE_SB_BufferD_t::Buffer, CFE_ES_GetAppID(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_EVS_EventType_ERROR, CFE_EVS↩_EventType_INFORMATION, CFE_EVS_SendEventWithAppID(), CFE_MISSION_SB_MAX_SB_MSG_SIZE, CF↩E_MISSION_SB_MSG_LIM_PERF_ID, CFE_MISSION_SB_PIPE_OFLOW_PERF_ID, CFE_PSP_GetProcessorId(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_BUF_ALOC_ERR, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_↩DecrBufUseCnt(), CFE_SB_FinishSendEvent(), CFE_SB_GET_BUF_ERR_EID, CFE_SB_GET_BUF_ERR_EID_BIT, CFE_SB_GetAppTskName(), CFE_SB_GetBufferFromCaller(), CFE_SB_GetBufferFromPool(), CFE_SB_GetMsgId(), CFE_SB_GetPipeName(), CFE_SB_GetPktType(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_GetTotalMsgLength(), CFE_SB_GRANTED, CFE_SB_INACTIVE, CFE_SB_INCREMENT_TLM, CFE_S↩B_IsValidMsgId(), CFE_SB_IsValidRouteIdx(), CFE_SB_LockSharedData(), CFE_SB_MSG_TOO_BIG, CFE_SB_↩MSG_TOO_BIG_EID, CFE_SB_MSGID_LIM_ERR_EID, CFE_SB_MSGID_LIM_ERR_EID_BIT, CFE_SB_PIPEO↩PTS_IGNOREMINE, CFE_SB_Q_FULL_ERR_EID, CFE_SB_Q_FULL_ERR_EID_BIT, CFE_SB_Q_WR_ERR_EID, CFE_SB_Q_WR_ERR_EID_BIT, CFE_SB_RequestToSendEvent(), CFE_SB_SEND_BAD_ARG_EID, CFE_SB_SE↩ND_INV_MSGID_EID, CFE_SB_SEND_NO_SUBS_EID, CFE_SB_SEND_NO_SUBS_EID_BIT, CFE_SB_SEND_Z↩EROCOPY, CFE_SB_SetMsgSeqCnt(), CFE_SB_TLM, CFE_SB_TLM_PIPEDEPTHSTATS_SIZE, CFE_SB_Unlock↩SharedData(), CFE_SUCCESS, CFE_SB_EventBuf_t::EvtsToSnd, cfe_sb_t::HKTlmMsg, CFE_SB_Housekeeping↩Tlm_Payload_t::InternalErrorCounter, CFE_SB_PipeDepthStats_t::InUse, CFE_SB_RouteEntry_t::MsgId, CFE_SB_↩HousekeepingTlm_Payload_t::MsgLimitErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::MsgSendErrorCounter,

CFE_SB_HousekeepingTlm_Payload_t::NoSubscribersCounter, NULL, CFE_SB_PipeD_t::Opts, OS_MAX_API_NA↩ ME, OS_QUEUE_FULL, OS_QueuePut(), OS_SUCCESS, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_PipeDepthStats_t::PeakInUse, CFE_SB_StatsTlm_Payload_t::PipeDepth↩ Stats, CFE_SB_HousekeepingTlm_Payload_t::PipeOverflowErrorCounter, cfe_sb_t::PipeTbl, CFE_SB_SenderId_t↩ ::ProcessorId, CFE_SB_BufferD_t::Sender, cfe_sb_t::SenderReporting, CFE_SB_PipeD_t::SendErrors, CFE_SB_↩ RouteEntry_t::SeqCnt, cfe_sb_t::StatTlmMsg, and CFE_SB_PipeD_t::SysQueueId.

Referenced by CFE_SB_PassMsg(), CFE_SB_SendMsg(), CFE_SB_ZeroCopyPass(), and CFE_SB_ZeroCopySend().

Here is the call graph for this function:



13.88.3.56 CFE_SB_SendPipeInfo()

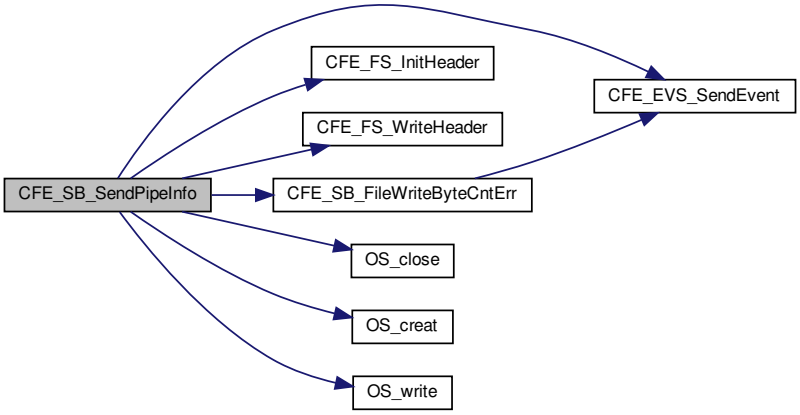
```
int32 CFE_SB_SendPipeInfo (
            const char * Filename )
```

Definition at line 945 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_Init↩ Header(), CFE_FS_SubType_SB_PIPEDATA, CFE_FS_WriteHeader(), CFE_PLATFORM_SB_MAX_PIPES, CFE↩ _SB_FILE_IO_ERR, CFE_SB_FileWriteByteCntErr(), CFE_SB_IN_USE, CFE_SB_SND_RTG_EID, CFE_SB_SND↩ _RTG_ERR1_EID, CFE_SUCCESS, CFE_SB_PipeD_t::InUse, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_↩ write(), OS_WRITE_ONLY, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_SendPipeInfoCmd().

Here is the call graph for this function:



13.88.3.57 CFE_SB_SendPipeInfoCmd()

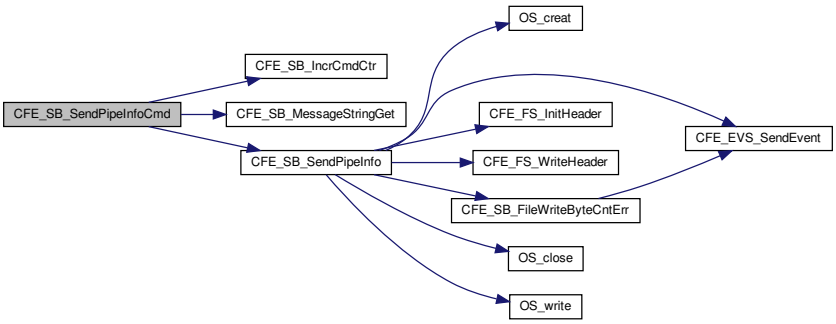
```
int32 CFE_SB_SendPipeInfoCmd (
            const CFE_SB_SendPipeInfo_t * data )
```

Definition at line 770 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME, CFE_SB_IncrCmdCtr(), CFE_SB_MessageString↩Get(), CFE_SB_SendPipeInfo(), CFE_SUCCESS, CFE_SB_WriteFileInfoCmd_Payload_t::Filename, OS_MAX_PAT↩H_LEN, and CFE_SB_WriteFileInfoCmd_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.58 CFE_SB_SendPrevSubsCmd()

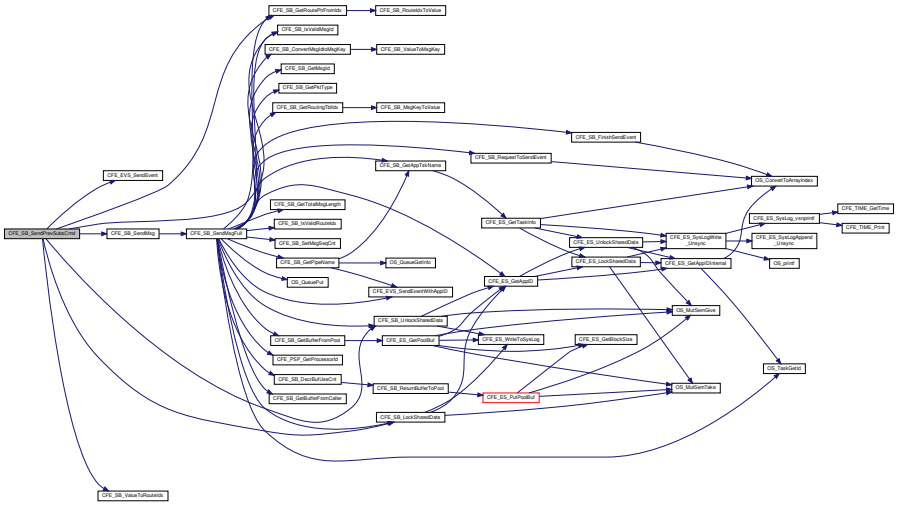
```
int32 CFE_SB_SendPrevSubsCmd (
            const CFE_SB_SendPrevSubs_t * data )
```

Definition at line 1101 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB_FULL_SUB_PKT_EID, CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GLOBAL, CFE_SB_IsValidMsgId(), CFE_SB_LockSharedData(), CFE_SB_PART_SUB_PKT_EID, CFE_SB_SendMsg(), CFE_SB_SUB_ENTRIES_PER_PKT, CFE_SB_UnlockSharedData(), CFE_SB_ValueToRouteIdx(), CFE_SUCCESS, CFE_SB_AllSubscriptionsTlm_Payload_t::Entries, CFE_SB_AllSubscriptionsTlm_Payload_t::Entry, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, CFE_SB_SubEntries_t::MsgId, NULL, CFE_SB_AllSubscriptionsTlm_t::Payload, CFE_SB_AllSubscriptionsTlm_Payload_t::PktSegment, cfe_sb_t::PrevSubMsg, CFE_SB_Qos_t::Priority, CFE_SB_SubEntries_t::Qos, CFE_SB_Qos_t::Reliability, cfe_sb_t::RoutingTbl, and CFE_SB_DestinationD_t::Scope.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.59 CFE_SB_SendRoutingInfoCmd()

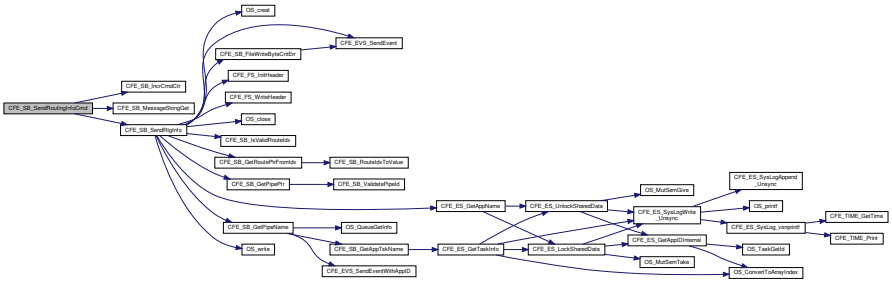
```
int32 CFE_SB_SendRoutingInfoCmd (
            const CFE_SB_SendRoutingInfo_t * data )
```

Definition at line 740 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME, CFE_SB_IncrCmdCtr(), CFE_SB_MessageStringGet(), CFE_SB_SendRtgInfo(), CFE_SUCCESS, CFE_SB_WriteFileInfoCmd_Payload_t::Filename, OS_MAX_PATH_LEN, and CFE_SB_WriteFileInfoCmd_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.60 CFE_SB_SendRtgInfo()

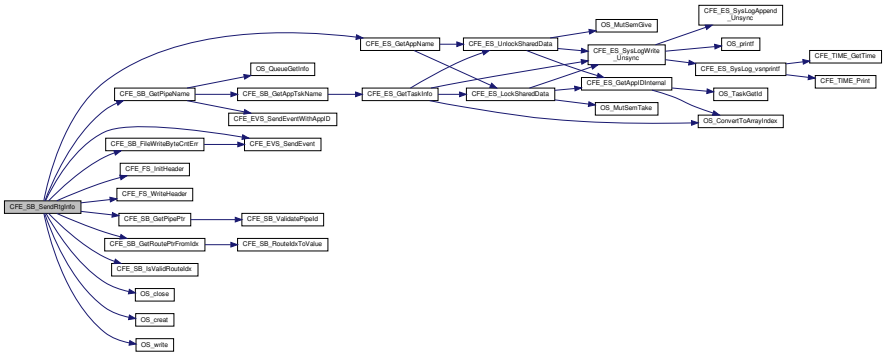
```
int32 CFE_SB_SendRtgInfo (
            const char * Filename )
```

Definition at line 831 of file cfe_sb_task.c.

References CFE_SB_PipeD_t::AppId, CFE_SB_RoutingFileEntry_t::AppName, CFE_ES_GetAppName(), CFE_EV↩
S_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_↩
SubType_SB_ROUTEDATA, CFE_FS_WriteHeader(), CFE_SB_FILE_IO_ERR, CFE_SB_FileWriteByteCntErr(), CF↩
E_SB_GetPipeName(), CFE_SB_GetPipePtr(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_IsValidRouteIdx(), CFE_↩
SB_MAX_NUMBER_OF_MSG_KEYS, CFE_SB_SND_RTG_EID, CFE_SB_SND_RTG_ERR1_EID, CFE_SUCCESS,
CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RoutingFileEntry_t::MsgCnt, CFE_SB_RouteEntry_t::MsgId, CFE_S↩
B_RoutingFileEntry_t::MsgId, cfe_sb_t::MsgMap, CFE_SB_DestinationD_t::Next, NULL, OS_close(), OS_creat(), O↩
S_FS_SUCCESS, OS_write(), OS_WRITE_ONLY, CFE_SB_RoutingFileEntry_t::PipeId, CFE_SB_RoutingFileEntry↩
_t::PipeName, and CFE_SB_RoutingFileEntry_t::State.

Referenced by CFE_SB_SendRoutingInfoCmd().

Here is the call graph for this function:



13.88.3.61 CFE_SB_SendStatsCmd()

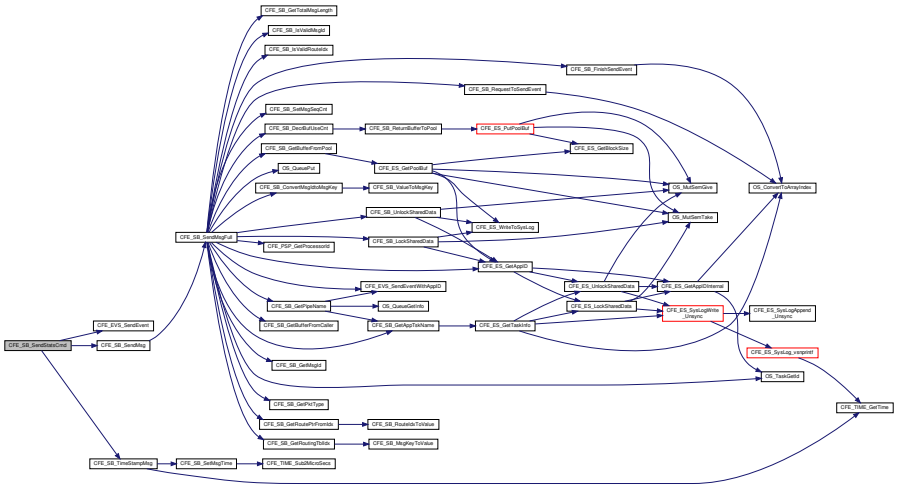
```
int32 CFE_SB_SendStatsCmd (
            const CFE_SB_SendSbStats_t * data )
```

Definition at line 713 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_SB_SendMsg(), CFE_SB_SND_STATS_EID, CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_t::Payload, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.88.3.62 CFE_SB_SetMsgSeqCnt()

```
void CFE_SB_SetMsgSeqCnt (
            CFE_SB_MsgPtr_t MsgPtr,
            uint32 Count )
```

Definition at line 552 of file cfe_sb_priv.c.

References CCSDS_WR_SEQ, and CFE_SB_Msg_t::Hdr.

Referenced by CFE_SB_SendMsgFull().

13.88.3.63 CFE_SB_SetRoutingTblIdx()

```
void CFE_SB_SetRoutingTblIdx (
             CFE_SB_MsgKey_t MsgKey,
             CFE_SB_MsgRouteIdx_t Value )
```

Definition at line 461 of file cfe_sb_priv.c.

References CFE_SB, CFE_SB_MsgKeyToValue(), and cfe_sb_t::MsgMap.

Referenced by CFE_SB_SubscribeFull().

Here is the call graph for this function:

13.88.3.64 CFE_SB_SetSubscriptionReporting()

```
void CFE_SB_SetSubscriptionReporting (
             uint32 state )
```

Definition at line 1298 of file cfe_sb_task.c.

References cfe_sb_t::SubscriptionReporting.

Referenced by CFE_SB_DisableSubReportingCmd(), and CFE_SB_EnableSubReportingCmd().

13.88.3.65 CFE_SB_SubscribeFull()

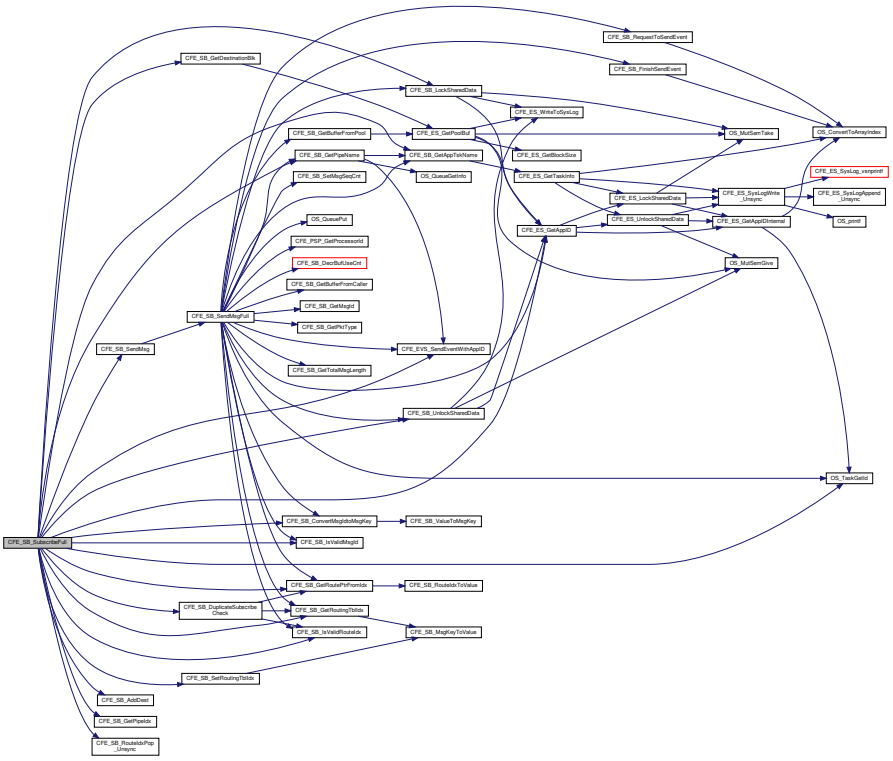
```
int32 CFE_SB_SubscribeFull (
             CFE_SB_MsgId_t MsgId,
             CFE_SB_PipeId_t PipeId,
             CFE_SB_Qos_t Quality,
             uint16 MsgLim,
             uint8 Scope )
```

Definition at line 867 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEventWithAppID(), CFE_PLATFORM_SB_MAX_DEST_PER_PKT, CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB, CFE_SB_ACTIVE, CFE_SB_AddDest(), CFE_SB_BAD_ARGUMENT, CFE_SB_BUF_ALOC_ERR, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_DEST_BLK_ERR_EID, CFE_SB_DUP_SUBSCRIP_EID, CFE_SB_DUPLICATE, CFE_SB_DuplicateSubscribeCheck(), CFE_SB_ENABLE, CFE_SB_GetAppTskName(), CFE_SB_GetDestinationBlk(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeName(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_GLOBAL, CFE_SB_INVALID_PIPE, CFE_SB_IsValidMsgId(), CFE_SB_IsValidRouteIdx(), CFE_SB_LockSharedData(), CFE_SB_MAX_DESTS_MET, CFE_SB_MAX_DESTS_MET_EID, CFE_SB_MAX_MSGS_MET, CFE_SB_MAX_MSGS_MET_EID, CFE_SB_RouteIdxPop_Unsync(), CFE_SB_SendMsg(), CFE_SB_SetRoutingTblIdx(), CFE_SB_SUB_ARG_ERR_EID, CFE_SB_SUB_INV_CALLER_EID, CFE_SB_SUB_INV_PIPE_EID, CFE_SB_SUBSCRIPTION, CFE_SB_SUBSCRIPTION_RCVD_EID, CFE_SB_SUBSCRIPTION_RPT_EID, CFE_SB_UnlockSharedData(), CFE_SUCCESS, CFE_SB_RouteEntry_t::Destinations, CFE_SB_HousekeepingTlm_Payload_t::DuplicateSubscriptionsCounter, cfe_sb_t::HKTlmMsg, CFE_SB_RouteEntry_t::MsgId, CFE_SB_SingleSubscriptionTlm_Payload_t::MsgId, CFE_SB_StatsTlm_Payload_t::MsgIdsInUse, NULL, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_SingleSubscriptionTlm_t::Payload, CFE_SB_StatsTlm_Payload_t::PeakMsgIdsInUse, CFE_SB_StatsTlm_Payload_t::PeakSubscriptionsInUse, CFE_SB_SingleSubscriptionTlm_Payload_t::Pipe, cfe_sb_t::PipeTbl, CFE_SB_Qos_t::Priority, CFE_SB_SingleSubscriptionTlm_Payload_t::Qos, CFE_SB_Qos_t::Reliability, cfe_sb_t::StatTlmMsg, cfe_sb_t::SubRprtMsg, CFE_SB_HousekeepingTlm_Payload_t::SubscribeErrorCounter, cfe_sb_t::SubscriptionReporting, CFE_SB_StatsTlm_Payload_t::SubscriptionsInUse, and CFE_SB_SingleSubscriptionTlm_Payload_t::SubType.

Referenced by CFE_SB_Subscribe(), CFE_SB_SubscribeEx(), and CFE_SB_SubscribeLocal().

Here is the call graph for this function:



13.88.3.66 CFE_SB_UnlockSharedData()

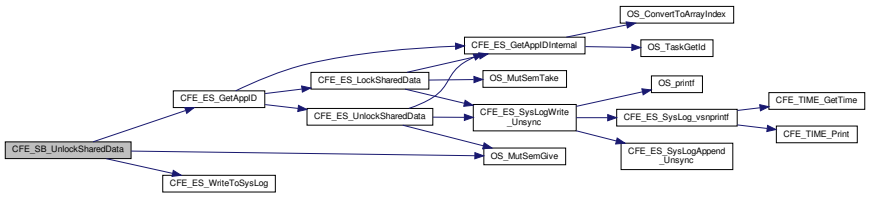
```
void CFE_SB_UnlockSharedData (
            const char * FuncName,
            int32 LineNumber )
```

Definition at line 317 of file cfe_sb_priv.c.

References CFE_ES_GetAppID(), CFE_ES_WriteToSysLog(), CFE_SB, OS_MutSemGive(), OS_SUCCESS, and cfe_sb_t::SharedDataMutexId.

Referenced by CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

Here is the call graph for this function:



13.88.3.67 CFE_SB_UnsubscribeFull()

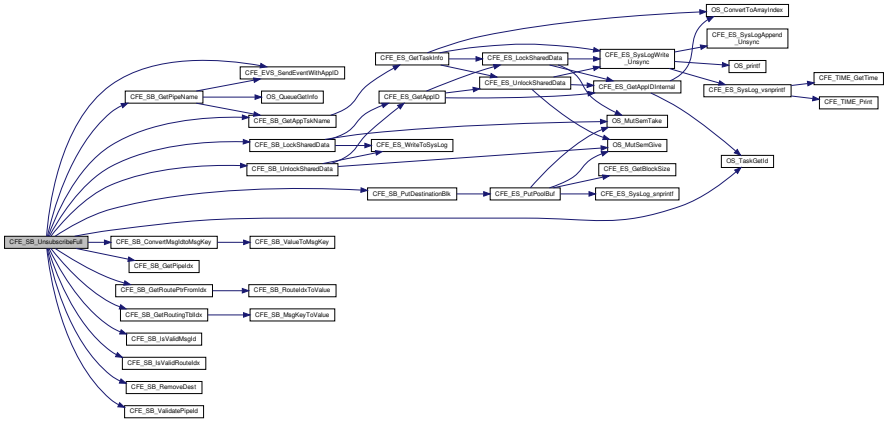
```
int32 CFE_SB_UnsubscribeFull (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            uint8 Scope,
            uint32 AppId )
```

Definition at line 1188 of file cfe_sb_api.c.

References CFE_SB_PipeD_t::AppId, cfe_sb_t::AppId, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEventWithAppID(), CFE_SB, CFE_SB_BAD_ARGUMENT, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_GetAppTskName(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeName(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_INVALID_PIPE, CFE_SB_IsValidMsgId(), CFE_SB_IsValidRouteIdx(), CFE_SB_LockSharedData(), CFE_SB_PutDestinationBlk(), CFE_SB_RemoveDest(), CFE_SB_SUBSCRIPTION_REMOVED_EID, CFE_SB_UnlockSharedData(), CFE_SB_UNSUB_ARG_ERR_EID, CFE_SB_UNSUB_INV_CALLER_EID, CFE_SB_UNSUB_INV_PIPE_EID, CFE_SB_UNSUB_NO_SUBS_EID, CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_RouteEntry_t::Destinations, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_DestinationD_t::Next, NULL, OS_MAX_API_NAME, OS_TaskGetId(), CFE_SB_StatsTlm_t::Payload, CFE_SB_DestinationD_t::PipeId, cfe_sb_t::PipeTbl, cfe_sb_t::StatTlmMsg, and CFE_SB_StatsTlm_Payload_t::SubscriptionsInUse.

Referenced by CFE_SB_Unsubscribe(), CFE_SB_UnsubscribeLocal(), and CFE_SB_UnsubscribeWithAppId().

Here is the call graph for this function:



13.88.3.68 CFE_SB_UnsubscribeWithAppId()

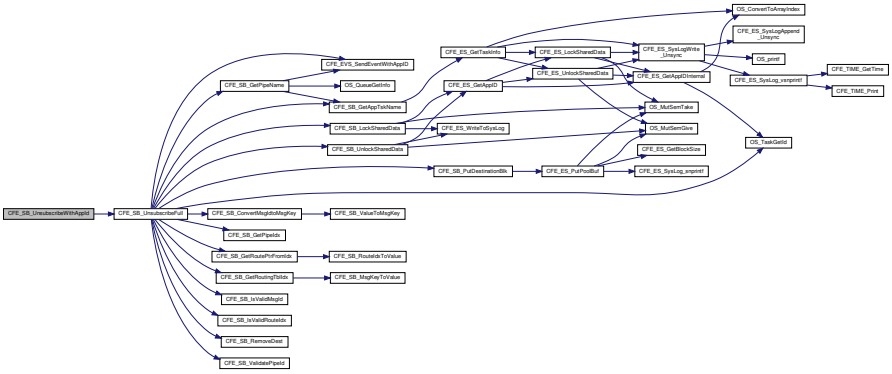
```
int32 CFE_SB_UnsubscribeWithAppId (
            CFE_SB_MsgId_t MsgId,
            CFE_SB_PipeId_t PipeId,
            uint32 AppId )
```

Definition at line 1147 of file cfe_sb_api.c.

References CFE_SB_LOCAL, and CFE_SB_UnsubscribeFull().

Referenced by CFE_SB_DeletePipeFull().

Here is the call graph for this function:



13.88.3.69 CFE_SB_ValidateMsgId()

```
int32 CFE_SB_ValidateMsgId (
            CFE_SB_MsgId_t MsgId )
```

Definition at line 572 of file cfe_sb_priv.c.

References CFE_SB_FAILED, CFE_SB_IsValidMsgId(), and CFE_SUCCESS.

Here is the call graph for this function:

13.88.3.70 CFE_SB_ValidatePipeId()

```
int32 CFE_SB_ValidatePipeId (
            CFE_SB_PipeId_t PipeId )
```

Definition at line 598 of file cfe_sb_priv.c.

References CFE_PLATFORM_SB_MAX_PIPES, CFE_SB, CFE_SB_FAILED, CFE_SB_NOT_IN_USE, CFE_SUCC↩ESS, CFE_SB_PipeD_t::InUse, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_EnableRouteCmd(), CFE_S↩B_GetLastSenderId(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_SetPipeOpts(), and CFE_SB_↩UnsubscribeFull().

13.88.3.71 CFE_SB_ValueToMsgKey()

```
static CFE_SB_MsgKey_t CFE_SB_ValueToMsgKey (
            CFE_SB_MsgKey_Atom_t KeyIdx )  [inline], [static]
```

Converts the supplied "bare number" into a type-safe CFE_SB_MsgKey_t value

Returns

CFE_SB_MsgKey_t value

Definition at line 536 of file cfe_sb_priv.h.

Referenced by CFE_SB_ConvertMsgIdtoMsgKey(), and CFE_SB_SendMapInfo().

13.88.3.72 CFE_SB_ValueToRouteIdx()

```
static CFE_SB_MsgRouteIdx_t CFE_SB_ValueToRouteIdx (
            CFE_SB_MsgRouteIdx_Atom_t TableIdx )  [inline], [static]
```

Converts the supplied "bare number" into a type-safe CFE_SB_MsgRouteIdx_t value

Returns

CFE_SB_MsgRouteIdx_t value

Definition at line 548 of file cfe_sb_priv.h.

Referenced by CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_InitIdxStack(), and CFE_SB_SendPrevSubsCmd().

13.88.3.73 CFE_SB_WriteQueue()

```
int32 CFE_SB_WriteQueue (
            CFE_SB_PipeD_t * pd,
            uint32 TskId,
            const CFE_SB_BufferD_t * bd,
            CFE_SB_MsgId_t MsgId )
```

13.88.3.74 CFE_SB_ZeroCopyReleaseAppId()

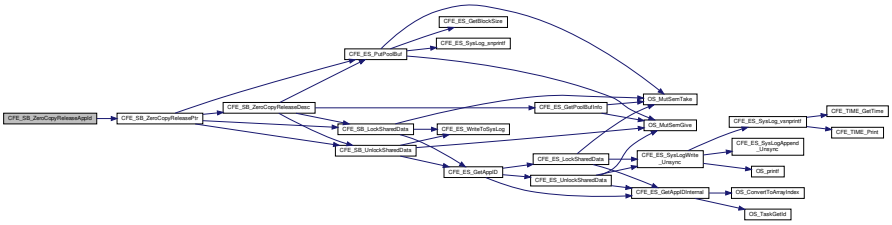
```
int32 CFE_SB_ZeroCopyReleaseAppId (
            uint32 AppId )
```

Definition at line 880 of file cfe_sb_priv.c.

References CFE_SB_ZeroCopyD_t::AppID, CFE_SB_ZeroCopyD_t::Buffer, CFE_SB, CFE_SB_ZeroCopyReleasePtr(), CFE_SUCCESS, NULL, CFE_SB_ZeroCopyD_t::Prev, and cfe_sb_t::ZeroCopyTail.

Referenced by CFE_SB_CleanUpApp().

Here is the call graph for this function:



13.88.3.75 CFE_SB_ZeroCopyReleaseDesc()

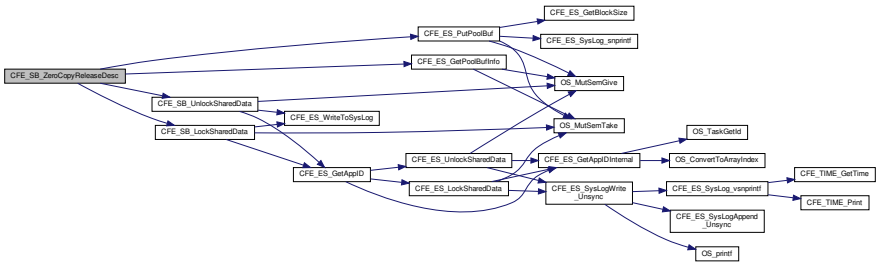
```
int32 CFE_SB_ZeroCopyReleaseDesc (
            CFE_SB_Msg_t * Ptr2Release,
            CFE_SB_ZeroCopyHandle_t BufferHandle )
```

Definition at line 2126 of file cfe_sb_api.c.

References CFE_ES_GetPoolBufInfo(), CFE_ES_PutPoolBuf(), CFE_SB, CFE_SB_BUFFER_INVALID, CFE_SB_↩ LockSharedData(), CFE_SB_UnlockSharedData(), CFE_SUCCESS, cfe_sb_t::Mem, CFE_SB_StatsTlm_Payload_t::↩ MemInUse, NULL, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, cfe_sb_t::StatTlmMsg, and cfe↩ _sb_t::ZeroCopyTail.

Referenced by CFE_SB_ZeroCopyPass(), CFE_SB_ZeroCopyReleasePtr(), and CFE_SB_ZeroCopySend().

Here is the call graph for this function:



13.88.4 Variable Documentation

13.88.4.1 CFE_SB

cfe_sb_t CFE_SB

Definition at line 49 of file cfe_sb_task.c.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_EarlyInit(), CFE_SB_FinishSendEvent(), CFE_SB_GetAvailPipeIdx(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_InitBuffers(), CFE_SB_InitIdxStack(), CFE_SB_InitMsgMap(), CFE_SB_InitPipeTbl(), CFE_SB_InitRoutingTbl(), CFE_SB_LockSharedData(), CFE_SB_PutDestinationBlk(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_RequestToSendEvent(), CFE_SB_ReturnBufferToPool(), CFE_SB_RouteIdxPop_Unsync(), CFE_SB_RouteIdxPush_Unsync(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SetRoutingTblIdx(), CFE_SB_SubscribeFull(), CFE_SB_UnlockSharedData(), CFE_SB_UnsubscribeFull(), CFE_SB_ValidatePipeId(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseAppId(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

13.89 cfe/fsw/cfe-core/src/sb/cfe_sb_task.c File Reference

```
#include "cfe_sb.h"
#include "cfe_sb_events.h"
#include "cfe_evs.h"
#include "cfe_sb_priv.h"
#include "osapi.h"
#include "cfe_version.h"
#include "cfe_msgids.h"
#include "cfe_error.h"
#include "cfe_es.h"
#include "cfe_psp.h"
#include "cfe_es_msg.h"
#include "cfe_sb_verify.h"
#include "cfe_sb_msg_id_util.h"
#include <string.h>
```

Functions

- void CFE_SB_TaskMain (void)

  Entry Point for cFE Core Application.
- int32 CFE_SB_AppInit (void)
- bool CFE_SB_VerifyCmdLength (CFE_SB_MsgPtr_t Msg, uint16 ExpectedLength)
- void CFE_SB_ProcessCmdPipePkt (void)
- int32 CFE_SB_NoopCmd (const CFE_SB_Noop_t *data)
- int32 CFE_SB_ResetCountersCmd (const CFE_SB_ResetCounters_t *data)
- int32 CFE_SB_EnableSubReportingCmd (const CFE_SB_EnableSubReporting_t *data)
- int32 CFE_SB_DisableSubReportingCmd (const CFE_SB_DisableSubReporting_t *data)
- int32 CFE_SB_SendHKTlmCmd (const CCSDS_CommandPacket_t *data)
- void CFE_SB_ResetCounters (void)
- int32 CFE_SB_EnableRouteCmd (const CFE_SB_EnableRoute_t *data)

- int32 CFE_SB_DisableRouteCmd (const CFE_SB_DisableRoute_t *data)
- int32 CFE_SB_SendStatsCmd (const CFE_SB_SendSbStats_t *data)
- int32 CFE_SB_SendRoutingInfoCmd (const CFE_SB_SendRoutingInfo_t *data)
- int32 CFE_SB_SendPipeInfoCmd (const CFE_SB_SendPipeInfo_t *data)
- int32 CFE_SB_SendMapInfoCmd (const CFE_SB_SendMapInfo_t *data)
- int32 CFE_SB_SendRtgInfo (const char *Filename)
- int32 CFE_SB_SendPipeInfo (const char *Filename)
- int32 CFE_SB_SendMapInfo (const char *Filename)
- int32 CFE_SB_SendPrevSubsCmd (const CFE_SB_SendPrevSubs_t *data)
- uint32 CFE_SB_FindGlobalMsgIdCnt (void)
- void CFE_SB_IncrCmdCtr (int32 status)
- void CFE_SB_FileWriteByteCntErr (const char *Filename, uint32 Requested, uint32 Actual)
- void CFE_SB_SetSubscriptionReporting (uint32 state)

Variables

- cfe_sb_t CFE_SB
- CFE_SB_Qos_t CFE_SB_Default_Qos

  Defines a default priority and reliabilty for off-board routing.

13.89.1 Function Documentation

13.89.1.1 CFE_SB_AppInit()

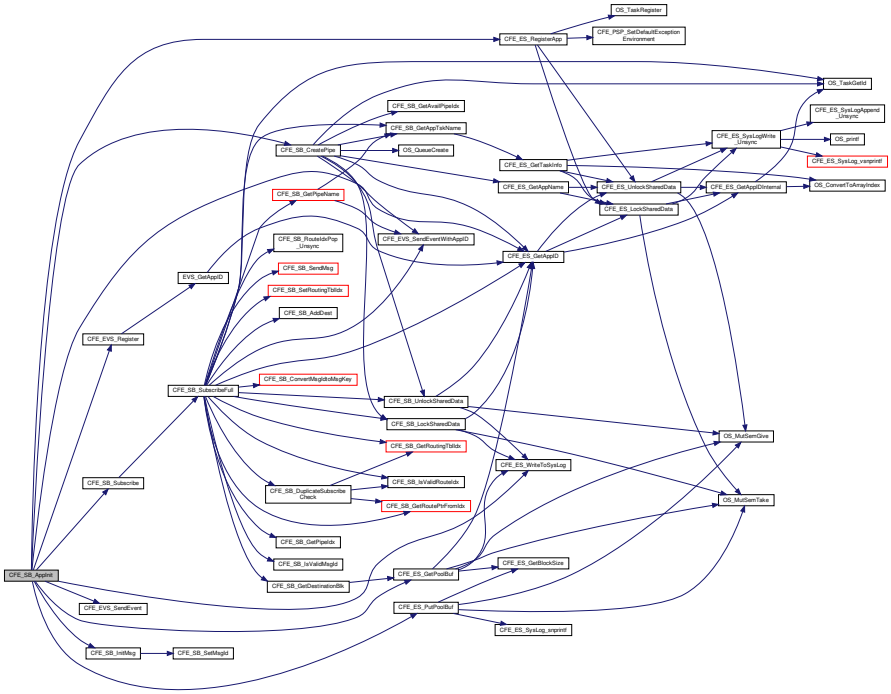
```
int32 CFE_SB_AppInit (
            void  )
```

Definition at line 136 of file cfe_sb_task.c.

References cfe_sb_t::AppId, CFE_ES_GetAppID(), CFE_ES_GetPoolBuf(), CFE_ES_PutPoolBuf(), CFE_ES_RegisterApp(), CFE_ES_WriteToSysLog(), CFE_EVS_EventFilter_BINARY, CFE_EVS_EventType_INFORMATION, CFE_EVS_Register(), CFE_EVS_SendEvent(), CFE_PLATFORM_EVS_MAX_EVENT_FILTERS, CFE_PLATFORM_SB_BUF_MEMORY_BYTES, CFE_PLATFORM_SB_FILTER_MASK1, CFE_PLATFORM_SB_FILTER_MASK2, CFE_PLATFORM_SB_FILTER_MASK3, CFE_PLATFORM_SB_FILTER_MASK4, CFE_PLATFORM_SB_FILTER_MASK5, CFE_PLATFORM_SB_FILTER_MASK6, CFE_PLATFORM_SB_FILTER_MASK7, CFE_PLATFORM_SB_FILTER_MASK8, CFE_PLATFORM_SB_FILTERED_EVENT1, CFE_PLATFORM_SB_FILTERED_EVENT2, CFE_PLATFORM_SB_FILTERED_EVENT3, CFE_PLATFORM_SB_FILTERED_EVENT4, CFE_PLATFORM_SB_FILTERED_EVENT5, CFE_PLATFORM_SB_FILTERED_EVENT6, CFE_PLATFORM_SB_FILTERED_EVENT7, CFE_PLATFORM_SB_FILTERED_EVENT8, CFE_PLATFORM_SB_MAX_DEST_PER_PKT, CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_PLATFORM_SB_MAX_PIPE_DEPTH, CFE_PLATFORM_SB_MAX_PIPES, CFE_SB_ALLSUBS_TLM_MID, CFE_SB_CMD_MID, CFE_SB_CMD_PIPE_DEPTH, CFE_SB_CMD_PIPE_NAME, CFE_SB_CreatePipe(), CFE_SB_HK_TLM_MID, CFE_SB_INIT_EID, CFE_SB_InitMsg(), CFE_SB_ONESUB_TLM_MID, CFE_SB_SEND_HK_MID, CFE_SB_SET_MEMADDR, CFE_SB_Subscribe(), CFE_SUCCESS, cfe_sb_t::CmdPipe, cfe_sb_t::EventFilters, CFE_EVS_BinFilter_t::EventID, cfe_sb_t::HKTlmMsg, CFE_EVS_BinFilter_t::Mask, CFE_SB_StatsTlm_Payload_t::MaxMemAllowed, CFE_SB_StatsTlm_Payload_t::MaxMsgIdsAllowed, CFE_SB_StatsTlm_Payload_t::MaxPipeDepthAllowed, CFE_SB_StatsTlm_Payload_t::MaxPipesAllowed, CFE_SB_StatsTlm_Payload_t::MaxSubscriptionsAllowed, cfe_sb_t::Mem, CFE_SB_HousekeepingTlm_Payload_t::MemPoolHandle, NULL, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_MemParams_t::PoolHdl, cfe_sb_t::PrevSubMsg, cfe_sb_t::StatTlmMsg, and cfe_sb_t::SubRprtMsg.

Referenced by CFE_SB_TaskMain().

Here is the call graph for this function:



13.89.1.2 CFE_SB_DisableRouteCmd()

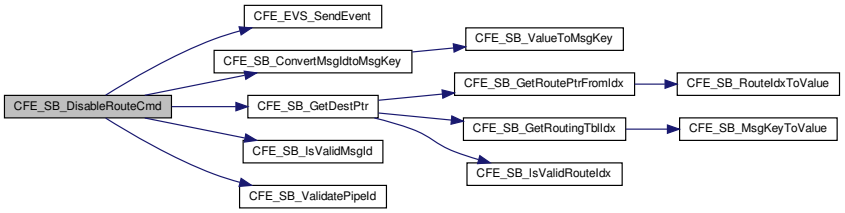
```
int32 CFE_SB_DisableRouteCmd (
            const CFE_SB_DisableRoute_t * data )
```

Definition at line 654 of file cfe_sb_task.c.

References CFE_SB_DestinationD_t::Active, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_DSBL_RTE1_EID, CFE_SB_DSBL_RTE2_EID, CFE_SB_DSBL_RTE3_EID, CFE_SB_GetDestPtr(), CFE_SB_INACTIVE, CFE_SB_IsValidMsgId(), CFE_SB_ValidatePipeId(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, CFE_SB_RouteCmd_Payload_t::MsgId, NULL, CFE_SB_RouteCmd_t::Payload, CFE_SB_HousekeepingTlm_t::Payload, and CFE_SB_RouteCmd_Payload_t::Pipe.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.3 CFE_SB_DisableSubReportingCmd()

```
int32 CFE_SB_DisableSubReportingCmd (
            const CFE_SB_DisableSubReporting_t * data )
```

Definition at line 513 of file cfe_sb_task.c.

References CFE_SB_DISABLE, CFE_SB_SetSubscriptionReporting(), and CFE_SUCCESS.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:

13.89.1.4 CFE_SB_EnableRouteCmd()

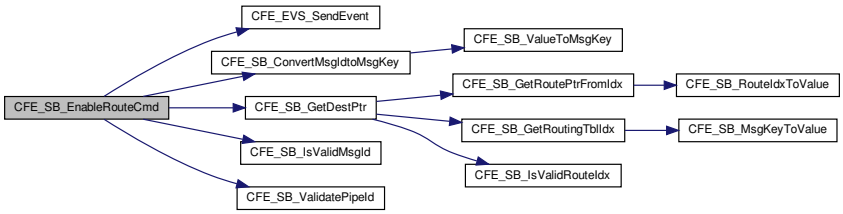
```
int32 CFE_SB_EnableRouteCmd (
            const CFE_SB_EnableRoute_t * data )
```

Definition at line 592 of file cfe_sb_task.c.

References CFE_SB_DestinationD_t::Active, CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE↩
_EVS_SendEvent(), CFE_SB_ACTIVE, CFE_SB_ConvertMsgIdtoMsgKey(), CFE_SB_ENBL_RTE1_EID, CFE_SB_↩
ENBL_RTE2_EID, CFE_SB_ENBL_RTE3_EID, CFE_SB_GetDestPtr(), CFE_SB_IsValidMsgId(), CFE_SB_Validate↩
PipeId(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_HousekeepingTlm↩
_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, CFE_SB_RouteCmd_Payload_t::MsgId, NULL, CFE_SB↩
_RouteCmd_t::Payload, CFE_SB_HousekeepingTlm_t::Payload, and CFE_SB_RouteCmd_Payload_t::Pipe.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.5 CFE_SB_EnableSubReportingCmd()

```
int32 CFE_SB_EnableSubReportingCmd (
            const CFE_SB_EnableSubReporting_t * data )
```

Definition at line 500 of file cfe_sb_task.c.

References CFE_SB_ENABLE, CFE_SB_SetSubscriptionReporting(), and CFE_SUCCESS.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:

13.89.1.6 CFE_SB_FileWriteByteCntErr()

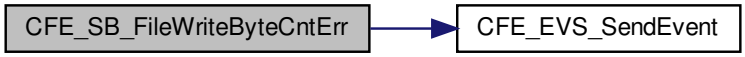
```
void CFE_SB_FileWriteByteCntErr (
            const char * Filename,
            uint32 Requested,
            uint32 Actual )
```

Definition at line 1277 of file cfe_sb_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), and CFE_SB_FILEWRITE_ERR_EID.

Referenced by CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

Here is the call graph for this function:



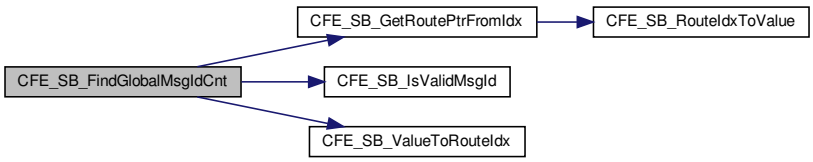
13.89.1.7 CFE_SB_FindGlobalMsgIdCnt()

```
uint32 CFE_SB_FindGlobalMsgIdCnt (
            void  )
```

Definition at line 1198 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GLOBAL, CFE_S↩B_IsValidMsgId(), CFE_SB_ValueToRouteIdx(), CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, NULL, and CFE_SB_DestinationD_t::Scope.

Here is the call graph for this function:



13.89.1.8 CFE_SB_IncrCmdCtr()

```
void CFE_SB_IncrCmdCtr (
            int32 status )
```

Definition at line 1253 of file cfe_sb_task.c.

References CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_Housekeeping↩Tlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

13.89.1.9 CFE_SB_NoopCmd()

```
int32 CFE_SB_NoopCmd (
            const CFE_SB_Noop_t * data )
```

Definition at line 466 of file cfe_sb_task.c.

References CFE_EVS_EventType_INFORMATION, CFE_EVS_SendEvent(), CFE_MAJOR_VERSION, CFE_MINOR_VERSION, CFE_MISSION_REV, CFE_REVISION, CFE_SB_CMD0_RCVD_EID, CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:

13.89.1.10 CFE_SB_ProcessCmdPipePkt()

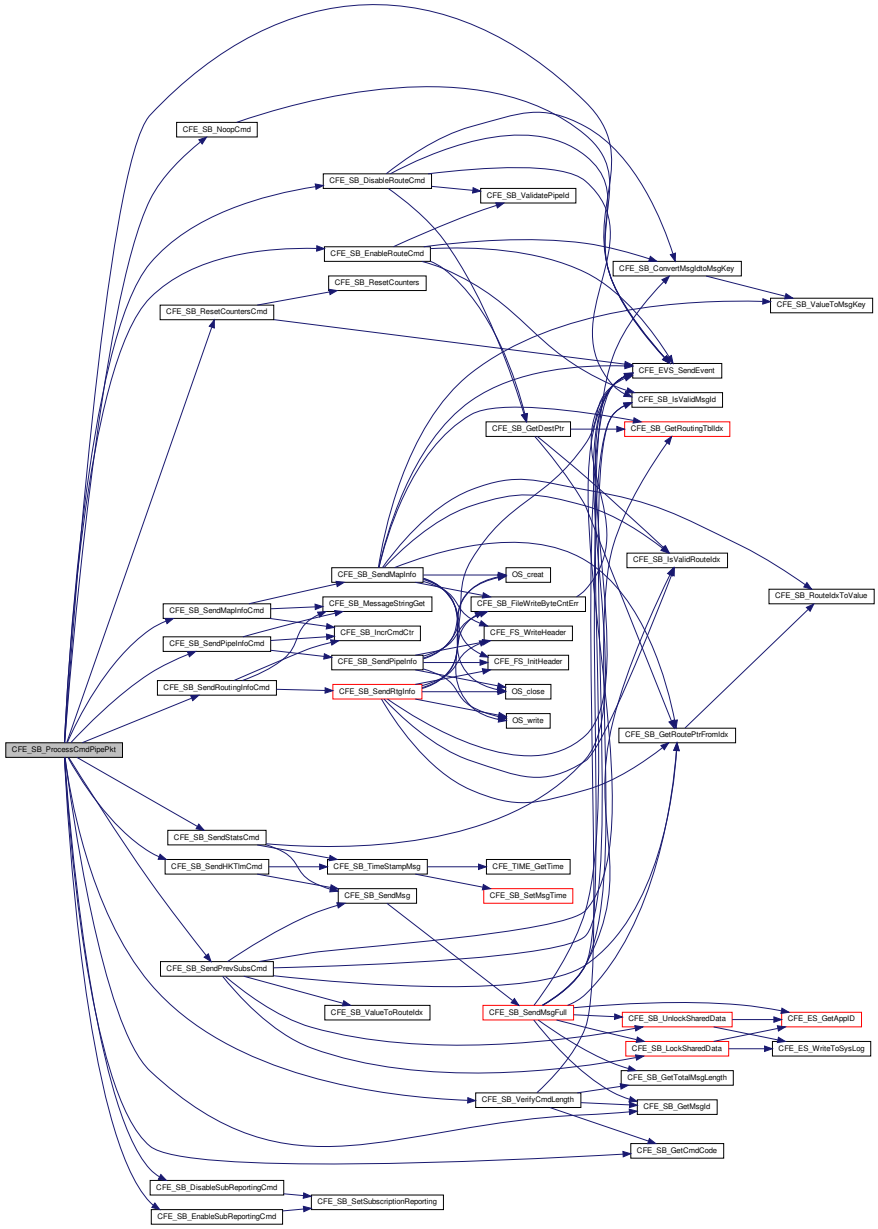
```
void CFE_SB_ProcessCmdPipePkt (
            void  )
```

Definition at line 349 of file cfe_sb_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_BAD_CMD_CODE_EID, CFE_SB_BAD_MSGID_EID, CFE_SB_CMD_MID, CFE_SB_DISABLE_ROUTE_CC, CFE_SB_DISABLE_SUB_REPORTING_CC, CFE_SB_DisableRouteCmd(), CFE_SB_DisableSubReportingCmd(), CFE_SB_ENABLE_ROUTE_CC, CFE_SB_ENABLE_SUB_REPORTING_CC, CFE_SB_EnableRouteCmd(), CFE_SB_EnableSubReportingCmd(), CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(), CFE_SB_NOOP_CC, CFE_SB_NoopCmd(), CFE_SB_RESET_COUNTERS_CC, CFE_SB_ResetCountersCmd(), CFE_SB_SEND_HK_MID, CFE_SB_SEND_MAP_INFO_CC, CFE_SB_SEND_PIPE_INFO_CC, CFE_SB_SEND_PREV_SUBS_CC, CFE_SB_SEND_ROUTING_INFO_CC, CFE_SB_SEND_SB_STATS_CC, CFE_SB_SendHKTlmCmd(), CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRoutingInfoCmd(), CFE_SB_SendStatsCmd(), CFE_SB_VerifyCmdLength(), cfe_sb_t::CmdPipePktPtr, CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_TaskMain().

Here is the call graph for this function:



13.89.1.11 CFE_SB_ResetCounters()

```
void CFE_SB_ResetCounters (
            void  )
```

Definition at line 562 of file cfe_sb_task.c.

References CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, CFE_SB_HousekeepingTlm_Payload_t::↩ CommandErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::CreatePipeErrorCounter, CFE_SB_Housekeeping↩ Tlm_Payload_t::DuplicateSubscriptionsCounter, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::↩ InternalErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::MsgLimitErrorCounter, CFE_SB_Housekeeping↩ Tlm_Payload_t::MsgReceiveErrorCounter, CFE_SB_HousekeepingTlm_Payload_t::MsgSendErrorCounter, CFE↩ _SB_HousekeepingTlm_Payload_t::NoSubscribersCounter, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_↩ HousekeepingTlm_Payload_t::PipeOverflowErrorCounter, and CFE_SB_HousekeepingTlm_Payload_t::Subscribe↩ ErrorCounter.

Referenced by CFE_SB_ResetCountersCmd().

13.89.1.12 CFE_SB_ResetCountersCmd()

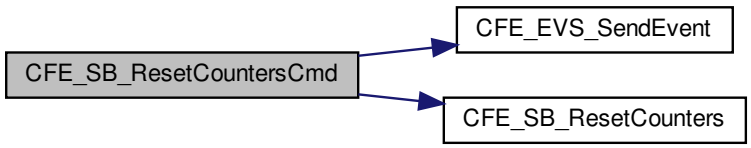
```
int32 CFE_SB_ResetCountersCmd (
            const CFE_SB_ResetCounters_t * data )
```

Definition at line 483 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_SB_CMD1_RCVD_EID, CFE_SB_Reset↩ Counters(), and CFE_SUCCESS.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.13 CFE_SB_SendHKTlmCmd()

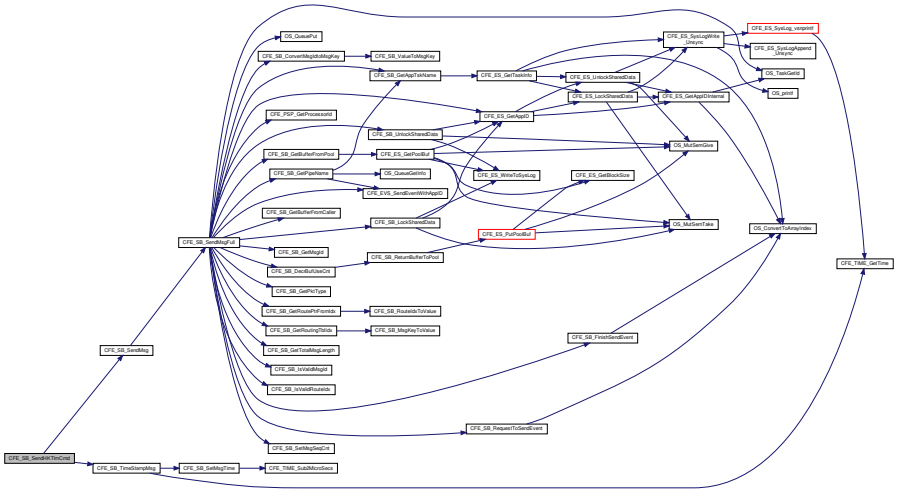
```
int32 CFE_SB_SendHKTlmCmd (
            const CCSDS_CommandPacket_t * data )
```

Definition at line 535 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_BUF_MEMORY_BYTES, CFE_SB_SendMsg(), CFE_SB_TimeStampMsg(), C↩ FE_SUCCESS, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_Payload_t::MemInUse, CFE_SB_StatsTlm_↩ Payload_t::MemInUse, CFE_SB_HousekeepingTlm_t::Payload, CFE_SB_StatsTlm_t::Payload, CFE_SB_StatsTlm_↩ Payload_t::PeakMemInUse, cfe_sb_t::StatTlmMsg, and CFE_SB_HousekeepingTlm_Payload_t::UnmarkedMem.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.14 CFE_SB_SendMapInfo()

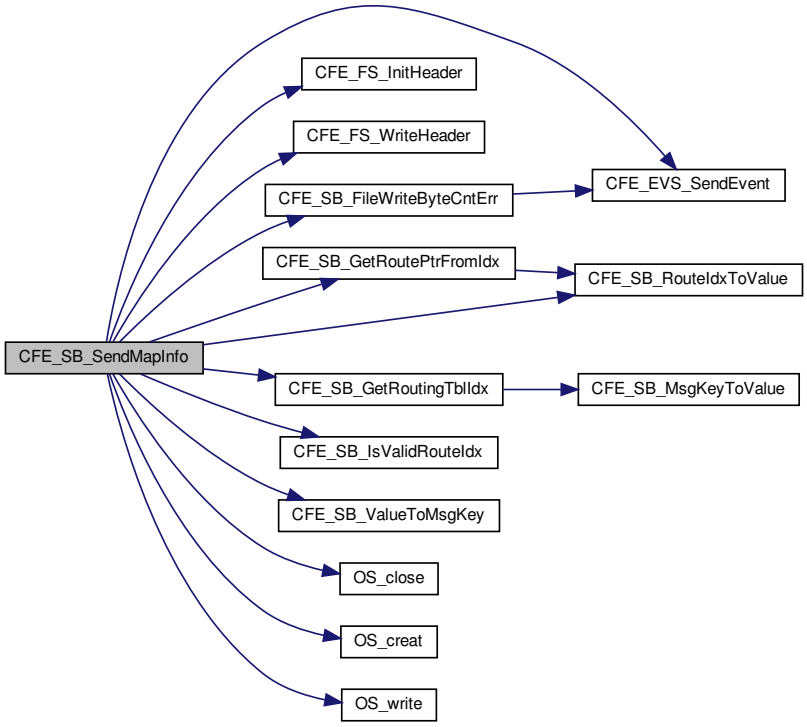
```
int32 CFE_SB_SendMapInfo (
            const char * Filename )
```

Definition at line 1017 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_Init↩
Header(), CFE_FS_SubType_SB_MAPDATA, CFE_FS_WriteHeader(), CFE_SB_FILE_IO_ERR, CFE_SB_FileWrite↩
ByteCntErr(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_IsValidRouteIdx(), CFE_SB_↩
MAX_NUMBER_OF_MSG_KEYS, CFE_SB_RouteIdxToValue(), CFE_SB_SND_RTG_EID, CFE_SB_SND_RTG_E↩
RR1_EID, CFE_SB_ValueToMsgKey(), CFE_SUCCESS, CFE_SB_MsgMapFileEntry_t::Index, CFE_SB_RouteEntry↩
_t::MsgId, CFE_SB_MsgMapFileEntry_t::MsgId, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_write(), and OS_↩
WRITE_ONLY.

Referenced by CFE_SB_SendMapInfoCmd().

Here is the call graph for this function:



13.89.1.15 CFE_SB_SendMapInfoCmd()

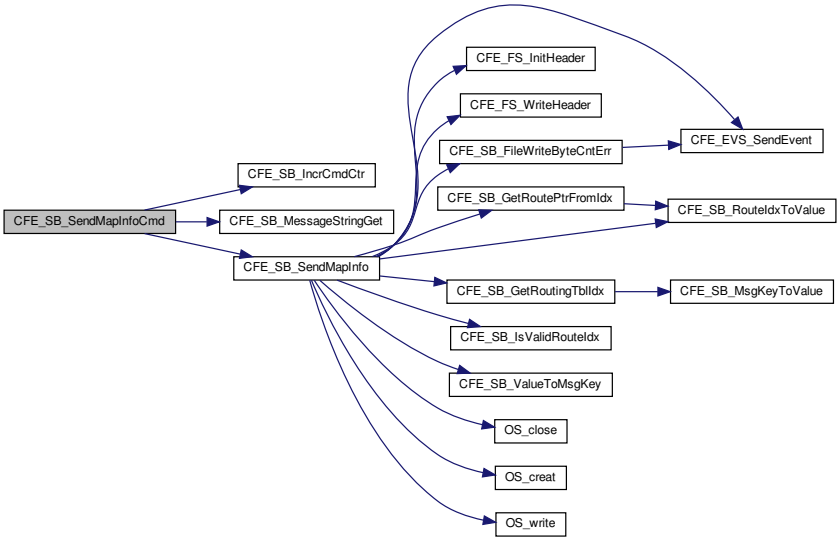
```
int32 CFE_SB_SendMapInfoCmd (
            const CFE_SB_SendMapInfo_t * data )
```

Definition at line 800 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME, CFE_SB_IncrCmdCtr(), CFE_SB_MessageString↩Get(), CFE_SB_SendMapInfo(), CFE_SUCCESS, CFE_SB_WriteFileInfoCmd_Payload_t::Filename, OS_MAX_PAT↩H_LEN, and CFE_SB_WriteFileInfoCmd_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.16 CFE_SB_SendPipeInfo()

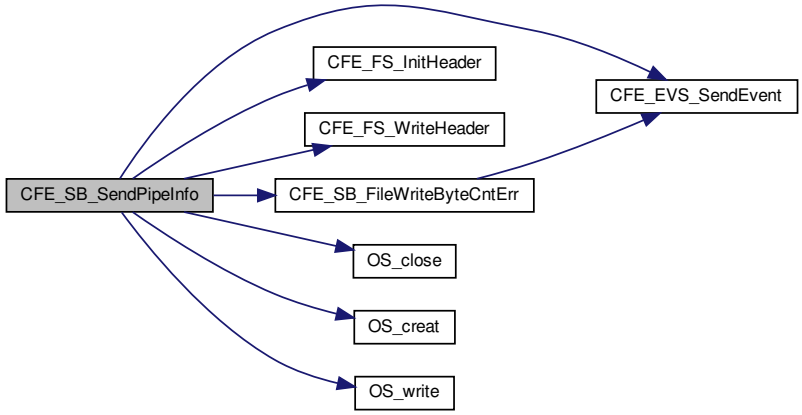
```
int32 CFE_SB_SendPipeInfo (
            const char * Filename )
```

Definition at line 945 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_SubType_SB_PIPEDATA, CFE_FS_WriteHeader(), CFE_PLATFORM_SB_MAX_PIPES, CFE_SB_FILE_IO_ERR, CFE_SB_FileWriteByteCntErr(), CFE_SB_IN_USE, CFE_SB_SND_RTG_EID, CFE_SB_SND_RTG_ERR1_EID, CFE_SUCCESS, CFE_SB_PipeD_t::InUse, OS_close(), OS_creat(), OS_FS_SUCCESS, OS_write(), OS_WRITE_ONLY, and cfe_sb_t::PipeTbl.

Referenced by CFE_SB_SendPipeInfoCmd().

Here is the call graph for this function:



13.89.1.17 CFE_SB_SendPipeInfoCmd()

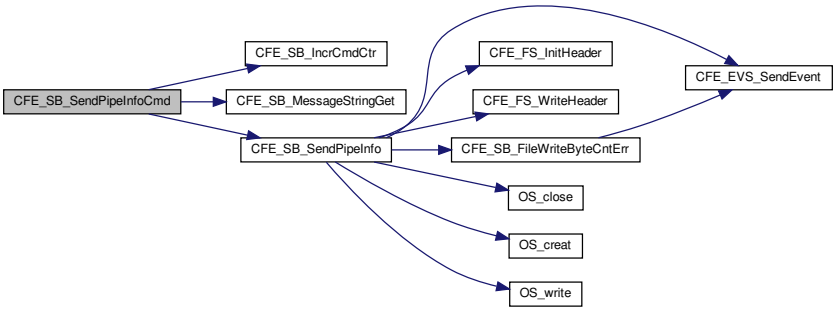
```
int32 CFE_SB_SendPipeInfoCmd (
            const CFE_SB_SendPipeInfo_t * data )
```

Definition at line 770 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME, CFE_SB_IncrCmdCtr(), CFE_SB_MessageString↩Get(), CFE_SB_SendPipeInfo(), CFE_SUCCESS, CFE_SB_WriteFileInfoCmd_Payload_t::Filename, OS_MAX_PAT↩H_LEN, and CFE_SB_WriteFileInfoCmd_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.18 CFE_SB_SendPrevSubsCmd()

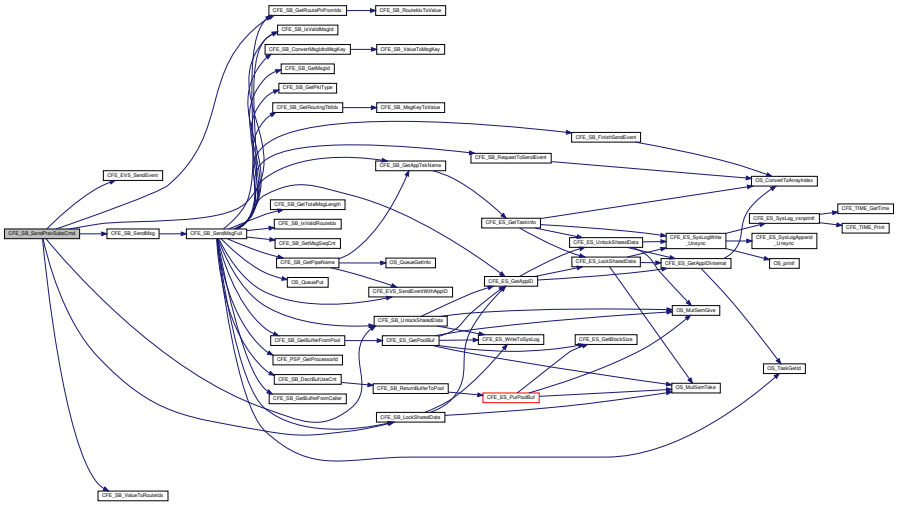
```
int32 CFE_SB_SendPrevSubsCmd (
            const CFE_SB_SendPrevSubs_t * data )
```

Definition at line 1101 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_PLATFORM_SB_MAX_MSG_IDS, CFE_SB_FULL_SUB_PKT_EID, CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GLOBAL, CFE_SB_IsValidMsgId(), CFE_SB_LockSharedData(), CFE_SB_PART_SUB_PKT_EID, CFE_SB_SendMsg(), CFE_SB_SUB_ENTRIES_PER_PKT, CFE_SB_UnlockSharedData(), CFE_SB_ValueToRouteIdx(), CFE_SUCCESS, CFE_SB_AllSubscriptionsTlm_Payload_t::Entries, CFE_SB_AllSubscriptionsTlm_Payload_t::Entry, CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RouteEntry_t::MsgId, CFE_SB_SubEntries_t::MsgId, NULL, CFE_SB_AllSubscriptionsTlm_t::Payload, CFE_SB_AllSubscriptionsTlm_Payload_t::PktSegment, cfe_sb_t::PrevSubMsg, CFE_SB_Qos_t::Priority, CFE_SB_SubEntries_t::Qos, CFE_SB_Qos_t::Reliability, cfe_sb_t::RoutingTbl, and CFE_SB_DestinationD_t::Scope.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.19 CFE_SB_SendRoutingInfoCmd()

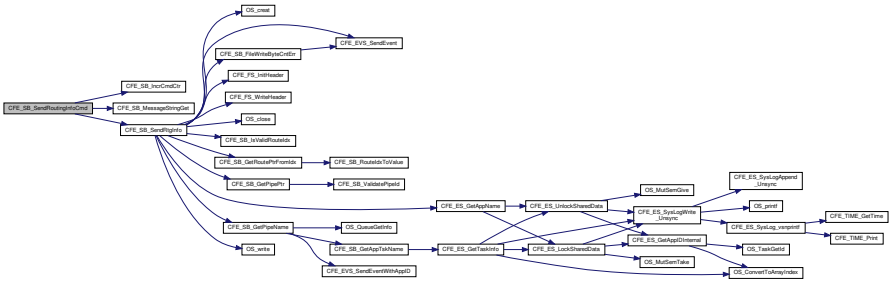
```
int32 CFE_SB_SendRoutingInfoCmd (
            const CFE_SB_SendRoutingInfo_t * data )
```

Definition at line 740 of file cfe_sb_task.c.

References CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME, CFE_SB_IncrCmdCtr(), CFE_SB_MessageStringGet(), CFE_SB_SendRtgInfo(), CFE_SUCCESS, CFE_SB_WriteFileInfoCmd_Payload_t::Filename, OS_MAX_PATH_LEN, and CFE_SB_WriteFileInfoCmd_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.20 CFE_SB_SendRtgInfo()

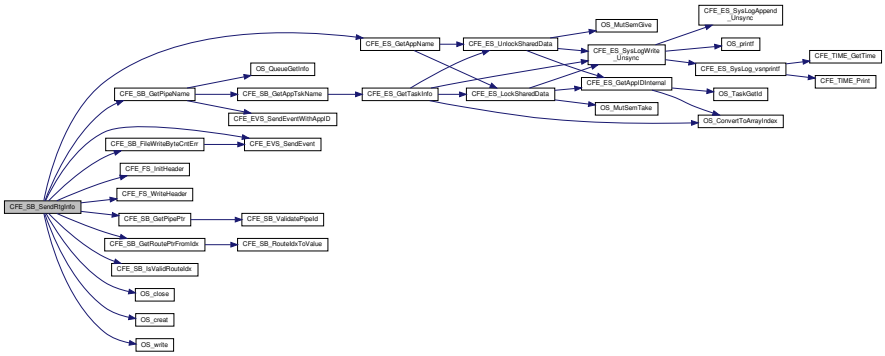
```
int32 CFE_SB_SendRtgInfo (
            const char * Filename )
```

Definition at line 831 of file cfe_sb_task.c.

References CFE_SB_PipeD_t::AppId, CFE_SB_RoutingFileEntry_t::AppName, CFE_ES_GetAppName(), CFE_EV↩
S_EventType_DEBUG, CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_FS_InitHeader(), CFE_FS_↩
SubType_SB_ROUTEDATA, CFE_FS_WriteHeader(), CFE_SB_FILE_IO_ERR, CFE_SB_FileWriteByteCntErr(), CF↩
E_SB_GetPipeName(), CFE_SB_GetPipePtr(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_IsValidRouteIdx(), CFE_↩
SB_MAX_NUMBER_OF_MSG_KEYS, CFE_SB_SND_RTG_EID, CFE_SB_SND_RTG_ERR1_EID, CFE_SUCCESS,
CFE_SB_RouteEntry_t::ListHeadPtr, CFE_SB_RoutingFileEntry_t::MsgCnt, CFE_SB_RouteEntry_t::MsgId, CFE_S↩
B_RoutingFileEntry_t::MsgId, cfe_sb_t::MsgMap, CFE_SB_DestinationD_t::Next, NULL, OS_close(), OS_creat(), O↩
S_FS_SUCCESS, OS_write(), OS_WRITE_ONLY, CFE_SB_RoutingFileEntry_t::PipeId, CFE_SB_RoutingFileEntry↩
_t::PipeName, and CFE_SB_RoutingFileEntry_t::State.

Referenced by CFE_SB_SendRoutingInfoCmd().

Here is the call graph for this function:



13.89.1.21 CFE_SB_SendStatsCmd()

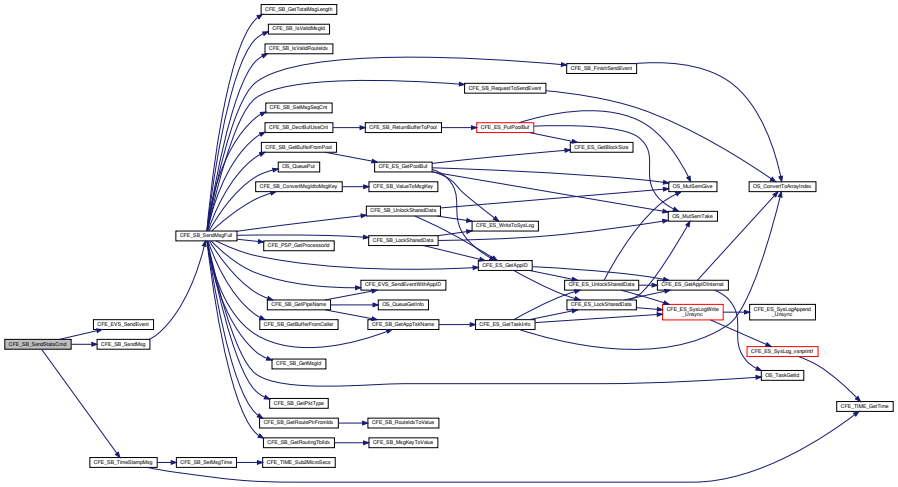
```
int32 CFE_SB_SendStatsCmd (
            const CFE_SB_SendSbStats_t * data )
```

Definition at line 713 of file cfe_sb_task.c.

References CFE_EVS_EventType_DEBUG, CFE_EVS_SendEvent(), CFE_SB_SendMsg(), CFE_SB_SND_STATS_EID, CFE_SB_TimeStampMsg(), CFE_SUCCESS, CFE_SB_HousekeepingTlm_Payload_t::CommandCounter, cfe_sb_t::HKTlmMsg, CFE_SB_HousekeepingTlm_t::Payload, and cfe_sb_t::StatTlmMsg.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.1.22 CFE_SB_SetSubscriptionReporting()

```
void CFE_SB_SetSubscriptionReporting (
            uint32 state )
```

Definition at line 1298 of file cfe_sb_task.c.

References cfe_sb_t::SubscriptionReporting.

Referenced by CFE_SB_DisableSubReportingCmd(), and CFE_SB_EnableSubReportingCmd().

13.89.1.23 CFE_SB_TaskMain()

```
void CFE_SB_TaskMain (
            void  )
```

Description

This is the entry point to the cFE SB Core Application.

Assumptions, External Events, and Notes:

None

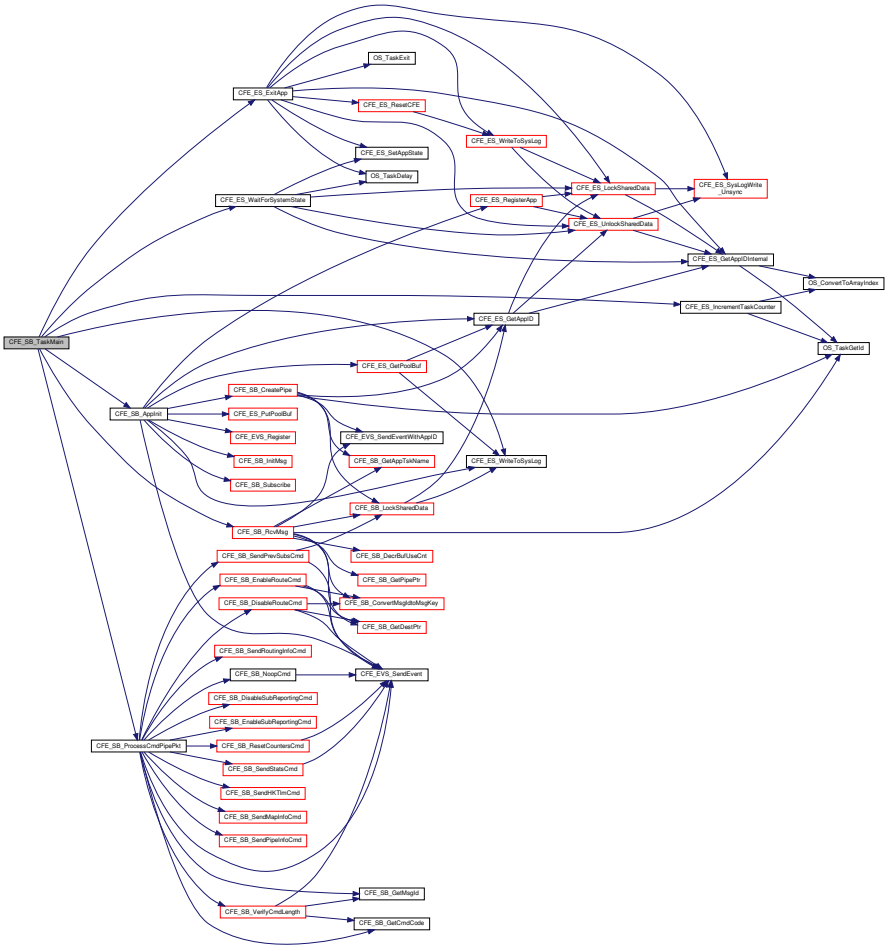
Return values

None	

Definition at line 65 of file cfe_sb_task.c.

References CFE_ES_ExitApp(), CFE_ES_IncrementTaskCounter(), CFE_ES_PerfLogEntry, CFE_ES_PerfLogExit, CFE_ES_RunStatus_CORE_APP_INIT_ERROR, CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR, CFE_ES↩
_SystemState_CORE_READY, CFE_ES_WaitForSystemState(), CFE_ES_WriteToSysLog(), CFE_MISSION_SB_M↩
AIN_PERF_ID, CFE_PLATFORM_CORE_MAX_STARTUP_MSEC, CFE_SB_AppInit(), CFE_SB_PEND_FOREVER, CFE_SB_ProcessCmdPipePkt(), CFE_SB_RcvMsg(), CFE_SUCCESS, cfe_sb_t::CmdPipe, and cfe_sb_t::CmdPipe↩
PktPtr.

Here is the call graph for this function:



13.89.1.24 CFE_SB_VerifyCmdLength()

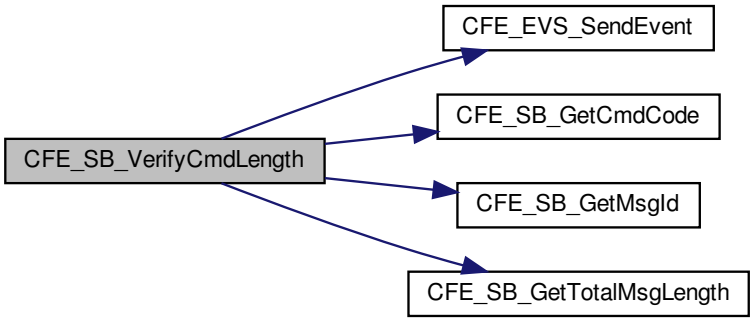
```
bool CFE_SB_VerifyCmdLength (
            CFE_SB_MsgPtr_t Msg,
            uint16 ExpectedLength )
```

Definition at line 311 of file cfe_sb_task.c.

References CFE_EVS_EventType_ERROR, CFE_EVS_SendEvent(), CFE_SB_GetCmdCode(), CFE_SB_GetMsgId(), CFE_SB_GetTotalMsgLength(), CFE_SB_LEN_ERR_EID, CFE_SB_HousekeepingTlm_Payload_t::CommandErrorCounter, cfe_sb_t::HKTlmMsg, and CFE_SB_HousekeepingTlm_t::Payload.

Referenced by CFE_SB_ProcessCmdPipePkt().

Here is the call graph for this function:



13.89.2 Variable Documentation

13.89.2.1 CFE_SB

```
cfe_sb_t CFE_SB
```

Definition at line 49 of file cfe_sb_task.c.

Referenced by CFE_SB_CleanUpApp(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_EarlyInit(), CFE_SB_FinishSendEvent(), CFE_SB_GetAvailPipeIdx(), CFE_SB_GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeIdx(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_GetRoutePtrFromIdx(), CFE_SB_GetRoutingTblIdx(), CFE_SB_InitBuffers(), CFE_SB_InitIdxStack(), CFE_SB_InitMsgMap(), CFE_SB_InitPipeTbl(), CFE_SB_InitRoutingTbl(), CFE_SB_LockSharedData(), CFE_SB_PutDestinationBlk(), CFE_SB_RcvMsg(), CFE_SB_ReadQueue(), CFE_SB_RequestToSendEvent(), CFE_SB_ReturnBufferToPool(), CFE_SB_RouteIdxPop_Unsync(), CFE_SB_RouteIdxPush_Unsync(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SetRoutingTblIdx(), CFE_SB_SubscribeFull(), CFE_SB_UnlockSharedData(), CFE_SB_UnsubscribeFull(), CFE_SB_ValidatePipeId(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyReleaseAppId(), CFE_SB_ZeroCopyReleaseDesc(), and CFE_SB_ZeroCopyReleasePtr().

13.89.2.2 CFE_SB_Default_Qos

```
CFE_SB_Qos_t CFE_SB_Default_Qos
```

Message Sender Identification Type Definition

Parameter used in CFE_SB_GetLastSenderId API which allows the receiver of a message to validate the sender of the message.

Definition at line 50 of file cfe_sb_task.c.

Referenced by CFE_ES_TaskInit(), CFE_EVS_TaskInit(), CFE_SB_EarlyInit(), CFE_SB_Subscribe(), and CFE_SB_↩ SubscribeLocal().

13.90 cfe/fsw/cfe-core/src/sb/cfe_sb_util.c File Reference

```
#include "cfe_sb.h"
#include "ccsds.h"
#include "osapi.h"
#include "cfe_error.h"
#include <string.h>
```

Functions

- void CFE_SB_InitMsg (void *MsgPtr, CFE_SB_MsgId_t MsgId, uint16 Length, bool Clear)

  Initialize a buffer for a software bus message.
- uint16 CFE_SB_MsgHdrSize (const CFE_SB_Msg_t *MsgPtr)

  Get the size of a software bus message header.
- void * CFE_SB_GetUserData (CFE_SB_MsgPtr_t MsgPtr)

  Get a pointer to the user data portion of a software bus message.
- uint16 CFE_SB_GetUserDataLength (const CFE_SB_Msg_t *MsgPtr)

  Gets the length of user data in a software bus message.
- void CFE_SB_SetUserDataLength (CFE_SB_MsgPtr_t MsgPtr, uint16 DataLength)

  Sets the length of user data in a software bus message.
- uint16 CFE_SB_GetTotalMsgLength (const CFE_SB_Msg_t *MsgPtr)

  Gets the total length of a software bus message.
- void CFE_SB_SetTotalMsgLength (CFE_SB_MsgPtr_t MsgPtr, uint16 TotalLength)

  Sets the total length of a software bus message.
- CFE_TIME_SysTime_t CFE_SB_GetMsgTime (CFE_SB_MsgPtr_t MsgPtr)

  Gets the time field from a software bus message.
- int32 CFE_SB_SetMsgTime (CFE_SB_MsgPtr_t MsgPtr, CFE_TIME_SysTime_t NewTime)

  Sets the time field in a software bus message.
- void CFE_SB_TimeStampMsg (CFE_SB_MsgPtr_t MsgPtr)

  Sets the time field in a software bus message with the current spacecraft time.
- uint16 CFE_SB_GetCmdCode (CFE_SB_MsgPtr_t MsgPtr)

  Gets the command code field from a software bus message.

- int32 CFE_SB_SetCmdCode (CFE_SB_MsgPtr_t MsgPtr, uint16 CmdCode)

  Sets the command code field in a software bus message.
- uint16 CFE_SB_GetChecksum (CFE_SB_MsgPtr_t MsgPtr)

  Gets the checksum field from a software bus message.
- void CFE_SB_GenerateChecksum (CFE_SB_MsgPtr_t MsgPtr)

  Calculates and sets the checksum of a software bus message.
- bool CFE_SB_ValidateChecksum (CFE_SB_MsgPtr_t MsgPtr)

  Validates the checksum of a software bus message.
- int32 CFE_SB_MessageStringGet (char ∗DestStringPtr, const char ∗SourceStringPtr, const char ∗DefaultString, uint32 DestMaxSize, uint32 SourceMaxSize)
- int32 CFE_SB_MessageStringSet (char ∗DestStringPtr, const char ∗SourceStringPtr, uint32 DestMaxSize, uint32 SourceMaxSize)

13.90.1 Function Documentation

13.90.1.1 CFE_SB_GenerateChecksum()

```
void CFE_SB_GenerateChecksum (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine calculates the checksum of a software bus message according to an implementation-defined algorithm. Then, it sets the checksum field in the message with the calculated value. The contents and location of this field will depend on the underlying implementation of software bus messages. It may be a checksum, a CRC, or some other algorithm.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a checksum field, then this routine will do nothing.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

See also

CFE_SB_ValidateChecksum, CFE_SB_GetChecksum

Definition at line 550 of file cfe_sb_util.c.

References CCSDS_LoadCheckSum(), CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, and CFE_SB_Msg↩_t::Hdr.

Here is the call graph for this function:

13.90.1.2 CFE_SB_GetChecksum()

```
uint16 CFE_SB_GetChecksum (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine gets the checksum (or other message integrity check value) from a software bus message. The contents and location of this field will depend on the underlying implementation of software bus messages. It may be a checksum, a CRC, or some other algorithm. Users should not call this function as part of a message integrity check (call CFE_SB_ValidateChecksum instead).

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a checksum field, then this routine will return a zero.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The checksum included in the software bus message header (if present), otherwise, returns a checksum value of zero.

Returns

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize, CFE_SB_ValidateChecksum, CFE_SB_GenerateChecksum

Definition at line 512 of file cfe_sb_util.c.

References CCSDS_RD_CHECKSUM, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CFE_SB_Msg_t::Hdr, and CCSDS_CommandPacket_t::Sec.

13.90.1.3 CFE_SB_GetCmdCode()

```
uint16 CFE_SB_GetCmdCode (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine gets the command code from a software bus message (if SB messages are implemented as CCSDS packets, this will be the function code).

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a command code field, then this routine will return a zero.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

Returns

The command code included in the software bus message header (if present). Otherwise, returns a command code value of zero.

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_SetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 443 of file cfe_sb_util.c.

References CCSDS_RD_FC, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CFE_SB_Msg_t::Hdr, and CCSDS_CommandPacket_t::Sec.

Referenced by CFE_ES_TaskPipe(), CFE_ES_VerifyCmdLength(), CFE_EVS_ProcessGroundCommand(), CFE_EVS_VerifyCmdLength(), CFE_SB_ProcessCmdPipePkt(), and CFE_SB_VerifyCmdLength().

13.90.1.4 CFE_SB_GetMsgTime()

```
CFE_TIME_SysTime_t CFE_SB_GetMsgTime (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine gets the time from a software bus message.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will return a zero time.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

Returns

The system time included in the software bus message header (if present), otherwise, returns a time value of zero.

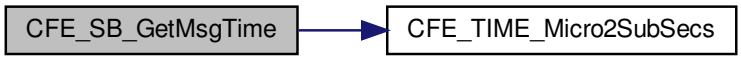
See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 290 of file cfe_sb_util.c.

References CCSDS_CMD, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CFE_TIME_Micro2SubSecs(), CFE_SB_Msg_t::Hdr, CCSDS_TelemetryPacket_t::Sec, CFE_TIME_SysTime_t::Seconds, CFE_TIME_SysTime_t::Subseconds, and CCSDS_TlmSecHdr_t::Time.

Here is the call graph for this function:



13.90.1.5 CFE_SB_GetTotalMsgLength()

```
uint16 CFE_SB_GetTotalMsgLength (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the total size of the software bus message.

Assumptions, External Events, and Notes:

- For the CCSDS implementation of this API, the size is derived from the message header.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

Returns

The total size (in bytes) of the software bus message, including headers.

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 242 of file cfe_sb_util.c.

References CCSDS_RD_LEN, and CFE_SB_Msg_t::Hdr.

Referenced by CFE_ES_VerifyCmdLength(), CFE_EVS_VerifyCmdLength(), CFE_SB_GetUserDataLength(), CFE_SB_SendMsgFull(), and CFE_SB_VerifyCmdLength().

13.90.1.6 CFE_SB_GetUserData()

```
void* CFE_SB_GetUserData (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine returns a pointer to the user data portion of a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function and avoid hard coding offsets into their SB message buffers.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message.

A pointer to the first byte of user data within the software bus message.

Returns

See also

CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 154 of file cfe_sb_util.c.

References CFE_SB_MsgHdrSize().

Here is the call graph for this function:

13.90.1.7 CFE_SB_GetUserDataLength()

```
uint16 CFE_SB_GetUserDataLength (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the size of the user data in a software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

The size (in bytes) of the user data in the software bus message.

Returns

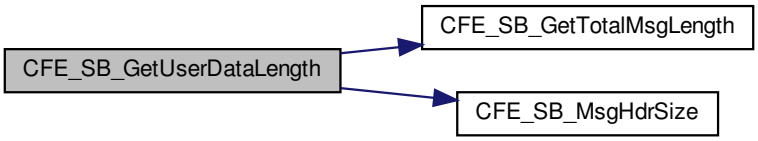
See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_SetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum, CFE_SB_MsgHdrSize

Definition at line 183 of file cfe_sb_util.c.

References CFE_SB_GetTotalMsgLength(), and CFE_SB_MsgHdrSize().

Here is the call graph for this function:



13.90.1.8 CFE_SB_InitMsg()

```
void CFE_SB_InitMsg (
            void * MsgPtr,
            CFE_SB_MsgId_t MsgId,
            uint16 Length,
            bool Clear )
```

Description

This routine fills in the header information needed to create a valid software bus message.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that will contain the message. This will point to the first byte of the message header. The `void*` data type allows the calling routine to use any data type when declaring its message buffer.
in	*MsgId*	The message ID to put in the message header.
in	*Length*	The total number of bytes of message data, including the SB message header .
in	*Clear*	A flag indicating whether to clear the rest of the message: • true - fill sequence count and packet data with zeroes. • false - leave sequence count and packet data unchanged.

See also

CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode

Definition at line 60 of file cfe_sb_util.c.

References CCSDS_CLR_PRI_HDR, CCSDS_INIT_SEQFLG, CCSDS_RD_SEQ, CCSDS_WR_LEN, CCSDS_WR_SEQ, CCSDS_WR_SEQFLG, CCSDS_WR_SHDR, and CFE_SB_SetMsgId().

Referenced by CFE_ES_TaskInit(), CFE_EVS_EarlyInit(), CFE_SB_AppInit(), CFE_SB_EarlyInit(), and EVS_GenerateEventTelemetry().

Here is the call graph for this function:

13.90.1.9 CFE_SB_MessageStringGet()

```
int32 CFE_SB_MessageStringGet (
            char * DestStringPtr,
            const char * SourceStringPtr,
            const char * DefaultString,
            uint32 DestMaxSize,
            uint32 SourceMaxSize )
```

Definition at line 612 of file cfe_sb_util.c.

References CFE_SB_BAD_ARGUMENT, and NULL.

Referenced by CFE_ES_AppCreate(), CFE_ES_DeleteCDSCmd(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_QueryOneCmd(), CFE_ES_ReloadAppCmd(), CFE_ES_RestartAppCmd(), CFE_ES_ShellCmd(), CFE_ES_StartAppCmd(), CFE_ES_StopAppCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_WriteERLogCmd(), CFE_ES_WriteSyslogCmd(), CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EVS_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EnableAppEventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_ResetAllFiltersCmd(), CFE_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfoCmd(), CFE_SB_SendPipeInfoCmd(), and CFE_SB_SendRoutingInfoCmd().

13.90.1.10 CFE_SB_MessageStringSet()

```
int32 CFE_SB_MessageStringSet (
            char * DestStringPtr,
            const char * SourceStringPtr,
            uint32 DestMaxSize,
            uint32 SourceMaxSize )
```

Definition at line 673 of file cfe_sb_util.c.

13.90.1.11 CFE_SB_MsgHdrSize()

```
uint16 CFE_SB_MsgHdrSize (
            const CFE_SB_Msg_t * MsgPtr )
```

Description

This routine returns the number of bytes in a software bus message header. This can be used for sizing buffers that need to store SB messages. SB message header formats can be different for each deployment of the cFE. So, applications should use this function and avoid hard coding their buffer sizes.

Assumptions, External Events, and Notes:

- For statically defined messages, a function call will not work. The macros CFE_SB_CMD_HDR_SIZE and CFE_SB_TLM_HDR_SIZE are available for use in static message buffer sizing or structure definitions.

Parameters

in	*MsgPtr	The message ID to calculate header size for. The size of the message header may depend on the MsgId in some implementations. For example, if SB messages are implemented as CCSDS packets, the size of the header is different for command vs. telemetry packets.

The number of bytes in the software bus message header for messages with the given `MsgId`. endstmt

Returns

See also

CFE_SB_GetUserData, CFE_SB_GetMsgId, CFE_SB_GetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_GetCmdCode, CFE_SB_GetChecksum

Definition at line 110 of file cfe_sb_util.c.

References CCSDS_CMD, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CFE_SB_CMD_HDR_SIZE, and CFE_SB_TLM_HDR_SIZE.

Referenced by CFE_SB_GetUserData(), CFE_SB_GetUserDataLength(), and CFE_SB_SetUserDataLength().

13.90.1.12 CFE_SB_SetCmdCode()

```
int32 CFE_SB_SetCmdCode (
            CFE_SB_MsgPtr_t MsgPtr,
            uint16 CmdCode )
```

Description

This routine sets the command code of a software bus message (if SB messages are implemented as CCSDS packets, this will be the function code).

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a command code field, then this routine will do nothing to the message contents and will return CFE_SB_WRONG_MSG_TYPE.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*CmdCode*	The command code to include in the message.

CFE_SUCCESS	Operation was performed successfully
CFE_SB_WRONG_MSG_TYPE	This error code will be returned when a request such as ...SetMsgTime is made on a packet that does not include a field for msg time.

Returns

See also

CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_GetCmdCode, CFE_SB_InitMsg

Definition at line 476 of file cfe_sb_util.c.

References CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CCSDS_WR_FC, CFE_SB_WRONG_MSG_TYPE, CFE_SUCCESS, CFE_SB_Msg_t::Hdr, and CCSDS_CommandPacket_t::Sec.

13.90.1.13 CFE_SB_SetMsgTime()

```
int32 CFE_SB_SetMsgTime (
            CFE_SB_MsgPtr_t MsgPtr,
            CFE_TIME_SysTime_t Time )
```

Description

This routine sets the time of a software bus message. Most applications will want to use CFE_SB_TimeStampMsg instead of this function. But, when needed, CFE_SB_SetMsgTime can be used to send a group of SB messages with identical time stamps.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will do nothing to the message contents and will return CFE_SB_WRONG_MSG_TYPE.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*Time*	The time to include in the message. This will usually be a time returned by the function CFE_TIME_GetTime().

CFE_SUCCESS	Operation was performed successfully
CFE_SB_WRONG_MSG_TYPE	This error code will be returned when a request such as ...SetMsgTime is made on a packet that does not include a field for msg time.

Returns

See also

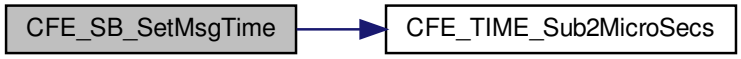
CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_GetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 358 of file cfe_sb_util.c.

References CCSDS_CMD, CCSDS_RD_SHDR, CCSDS_RD_TYPE, CFE_SB_WRONG_MSG_TYPE, CFE_SUCCESS, CFE_TIME_Sub2MicroSecs(), CFE_SB_Msg_t::Hdr, CCSDS_TelemetryPacket_t::Sec, CFE_TIME_SysTime_t::Seconds, CFE_TIME_SysTime_t::Subseconds, and CCSDS_TlmSecHdr_t::Time.

Referenced by CFE_SB_TimeStampMsg(), and EVS_GenerateEventTelemetry().

Here is the call graph for this function:



13.90.1.14 CFE_SB_SetTotalMsgLength()

```
void CFE_SB_SetTotalMsgLength (
             CFE_SB_MsgPtr_t MsgPtr,
             uint16 TotalLength )
```

Description

This routine sets the field in the SB message header that determines the total length of the message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function rather than trying to poke a length value directly into their SB message buffers.

Assumptions, External Events, and Notes:

None

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*TotalLength*	The length to set (total size of the message, in bytes, including headers).

See also

CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_GetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 267 of file cfe_sb_util.c.

References CCSDS_WR_LEN, and CFE_SB_Msg_t::Hdr.

13.90.1.15 CFE_SB_SetUserDataLength()

```
void CFE_SB_SetUserDataLength (
            CFE_SB_MsgPtr_t MsgPtr,
            uint16 DataLength )
```

Description

This routine sets the field in the SB message header that determines the size of the user data in a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function rather than trying to poke a length value directly into their SB message buffers.

Assumptions, External Events, and Notes:

- You must set a valid message ID in the SB message header before calling this function.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.
in	*DataLength*	The length to set (size of the user data, in bytes).

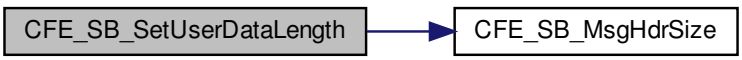
See also

CFE_SB_SetMsgId, CFE_SB_GetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_TimeStampMsg, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 214 of file cfe_sb_util.c.

References CCSDS_WR_LEN, CFE_SB_MsgHdrSize(), and CFE_SB_Msg_t::Hdr.

Here is the call graph for this function:



13.90.1.16 CFE_SB_TimeStampMsg()

```
void CFE_SB_TimeStampMsg (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine sets the time of a software bus message with the current spacecraft time. This will be the same time that is returned by the function CFE_TIME_GetTime.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will do nothing.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

See also

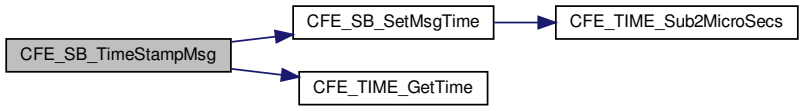
CFE_SB_SetMsgId, CFE_SB_SetUserDataLength, CFE_SB_SetTotalMsgLength, CFE_SB_SetMsgTime, CFE_SB_SetCmdCode, CFE_SB_InitMsg

Definition at line 424 of file cfe_sb_util.c.

References CFE_SB_SetMsgTime(), and CFE_TIME_GetTime().

Referenced by CFE_ES_HousekeepingCmd(), CFE_ES_QueryOneCmd(), CFE_ES_SendMemPoolStatsCmd(), CFE_ES_ShellOutputCommand(), CFE_EVS_ReportHousekeepingCmd(), CFE_SB_SendHKTlmCmd(), and CFE_SB_SendStatsCmd().

Here is the call graph for this function:



13.90.1.17 CFE_SB_ValidateChecksum()

```
bool CFE_SB_ValidateChecksum (
            CFE_SB_MsgPtr_t MsgPtr )
```

Description

This routine calculates the expected checksum of a software bus message according to an implementation-defined algorithm. Then, it checks the calculated value against the value in the message's checksum. If the checksums do not match, this routine will generate an event message reporting the error.

Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a checksum field, then this routine will always return `true`.

Parameters

in	*MsgPtr*	A pointer to the buffer that contains the software bus message. This must point to the first byte of the message header.

true	The checksum field in the packet is valid.
false	The checksum field in the packet is not valid or the message type is wrong.

Returns

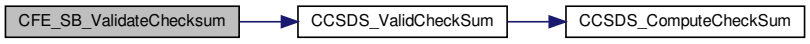
See also

CFE_SB_GenerateChecksum, CFE_SB_GetChecksum

Definition at line 581 of file cfe_sb_util.c.

References CCSDS_RD_SHDR, CCSDS_RD_TYPE, CCSDS_TLM, CCSDS_ValidCheckSum(), and CFE_SB_Msg↩_t::Hdr.

Here is the call graph for this function:



13.91 cfe/fsw/cfe-core/src/sb/cfe_sb_verify.h File Reference

```
#include <stdint.h>
```

13.92 osal/src/os/inc/common_types.h File Reference

```
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
```

Macros

- #define CompileTimeAssert(Condition, Message) typedef char Message[(Condition) ? 1 : -1]
- #define _EXTENSION_
- #define OS_PACK
- #define OS_ALIGN(n)
- #define OS_USED
- #define OS_PRINTF(n, m)
- #define TRUE true
- #define FALSE false
- #define NULL ((void *) 0)

Typedefs

- typedef int8_t int8
- typedef int16_t int16
- typedef int32_t int32
- typedef int64_t int64
- typedef uint8_t uint8
- typedef uint16_t uint16
- typedef uint32_t uint32
- typedef uint64_t uint64
- typedef intptr_t intptr
- typedef uintptr_t cpuaddr
- typedef size_t cpusize
- typedef ptrdiff_t cpudiff
- typedef bool osalbool
- typedef osalbool boolean

Functions

- CompileTimeAssert (sizeof(uint8)==1, TypeUint8WrongSize)
- CompileTimeAssert (sizeof(uint16)==2, TypeUint16WrongSize)
- CompileTimeAssert (sizeof(uint32)==4, TypeUint32WrongSize)
- CompileTimeAssert (sizeof(uint64)==8, TypeUint64WrongSize)
- CompileTimeAssert (sizeof(int8)==1, Typeint8WrongSize)
- CompileTimeAssert (sizeof(int16)==2, Typeint16WrongSize)
- CompileTimeAssert (sizeof(int32)==4, Typeint32WrongSize)
- CompileTimeAssert (sizeof(int64)==8, Typeint64WrongSize)
- CompileTimeAssert (sizeof(cpuaddr) >=sizeof(void *), TypePtrWrongSize)

13.92.1 Macro Definition Documentation

13.92.1.1 _EXTENSION_

```
#define _EXTENSION_
```

Definition at line 65 of file common_types.h.

13.92.1.2 CompileTimeAssert

```
#define CompileTimeAssert(
              Condition,
              Message ) typedef char Message[(Condition) ?  1 :  -1]
```

Definition at line 44 of file common_types.h.

13.92.1.3 FALSE

```
#define FALSE false
```

Definition at line 127 of file common_types.h.

13.92.1.4 NULL

```
#define NULL ((void *) 0)
```

Definition at line 135 of file common_types.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteCDS(), CFE_ES_DeleteCDSCmd(), CFE_ES_GetBlockSize(), CFE_ES_GetGenCount(), CFE_ES_GetGenCounterIDBy↩ Name(), CFE_ES_GetMemPoolStats(), CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_GetResetType(), CFE_ES_LoadLibrary(), CFE_ES_ParseFileEntry(), CFE_ES_PoolCreateEx(), CFE_ES_PutPoolBuf(), CFE_ES_↩ QueryOneCmd(), CFE_ES_RegisterCDSEx(), CFE_ES_RegisterGenCounter(), CFE_ES_ReloadAppCmd(), CFE↩ _ES_ResetCFE(), CFE_ES_RestartAppCmd(), CFE_ES_SetupResetVariables(), CFE_ES_ShellCmd(), CFE_ES_↩ ShellOutputCommand(), CFE_ES_StartAppCmd(), CFE_ES_StopAppCmd(), CFE_ES_TaskInit(), CFE_ES_Validate↩ Handle(), CFE_ES_WriteToERLog(), CFE_EVS_AddEventFilterCmd(), CFE_EVS_DeleteEventFilterCmd(), CFE_EV↩ S_DisableAppEventsCmd(), CFE_EVS_DisableAppEventTypeCmd(), CFE_EVS_EarlyInit(), CFE_EVS_EnableApp↩ EventsCmd(), CFE_EVS_EnableAppEventTypeCmd(), CFE_EVS_Register(), CFE_EVS_ResetAllFiltersCmd(), CF↩ E_EVS_ResetAppCounterCmd(), CFE_EVS_ResetFilter(), CFE_EVS_ResetFilterCmd(), CFE_EVS_SetFilterCmd(), CFE_EVS_TaskInit(), CFE_PSP_GetCDSSize(), CFE_PSP_GetCFETextSegmentInfo(), CFE_PSP_GetKernelText↩ SegmentInfo(), CFE_PSP_GetResetArea(), CFE_PSP_GetUserReservedArea(), CFE_PSP_GetVolatileDiskMem(), CFE_PSP_ReadFromCDS(), CFE_PSP_SetupLocal1Hz(), CFE_PSP_WriteToCDS(), CFE_SB_AddDest(), CFE_S↩ B_AppInit(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_DisableRouteCmd(), CFE_SB_Duplicate↩ SubscribeCheck(), CFE_SB_EarlyInit(), CFE_SB_EnableRouteCmd(), CFE_SB_FindGlobalMsgIdCnt(), CFE_SB_↩ GetBufferFromPool(), CFE_SB_GetDestinationBlk(), CFE_SB_GetDestPtr(), CFE_SB_GetLastSenderId(), CFE_S↩ B_GetPipeIdByName(), CFE_SB_GetPipeName(), CFE_SB_GetPipeOpts(), CFE_SB_GetPipePtr(), CFE_SB_Init↩ PipeTbl(), CFE_SB_InitRoutingTbl(), CFE_SB_MessageStringGet(), CFE_SB_PutDestinationBlk(), CFE_SB_Rcv↩ Msg(), CFE_SB_RemoveDest(), CFE_SB_SendMsgFull(), CFE_SB_SendPrevSubsCmd(), CFE_SB_SendRtgInfo(), CFE_SB_SubscribeFull(), CFE_SB_UnsubscribeFull(), CFE_SB_ZeroCopyGetPtr(), CFE_SB_ZeroCopyRelease↩ AppId(), CFE_SB_ZeroCopyReleaseDesc(), EVS_FindEventID(), EVS_GetApplicationInfo(), EVS_IsFiltered(), and main().

13.92.1.5 OS_ALIGN

```
#define OS_ALIGN(
              n )
```

Definition at line 67 of file common_types.h.

13.92.1.6 OS_PACK

```
#define OS_PACK
```

Definition at line 66 of file common_types.h.

13.92.1.7 OS_PRINTF

```
#define OS_PRINTF(
                n,
                m )
```

Definition at line 69 of file common_types.h.

13.92.1.8 OS_USED

```
#define OS_USED
```

Definition at line 68 of file common_types.h.

13.92.1.9 TRUE

```
#define TRUE true
```

Definition at line 123 of file common_types.h.

13.92.2 Typedef Documentation

13.92.2.1 boolean

```
typedef osalbool boolean
```

Definition at line 119 of file common_types.h.

13.92.2.2 cpuaddr

```
typedef uintptr_t cpuaddr
```

Definition at line 90 of file common_types.h.

13.92.2.3 cpudiff

```
typedef ptrdiff_t cpudiff
```

Definition at line 92 of file common_types.h.

13.92.2.4 cpusize

```
typedef size_t cpusize
```

Definition at line 91 of file common_types.h.

13.92.2.5 int16

```
typedef int16_t int16
```

Definition at line 82 of file common_types.h.

13.92.2.6 int32

```
typedef int32_t int32
```

Definition at line 83 of file common_types.h.

13.92.2.7 int64

```
typedef int64_t int64
```

Definition at line 84 of file common_types.h.

13.92.2.8 int8

```
typedef int8_t int8
```

Definition at line 81 of file common_types.h.

13.92.2.9 intptr

```
typedef intptr_t intptr
```

Definition at line 89 of file common_types.h.

13.92.2.10 osalbool

```
typedef bool osalbool
```

Definition at line 100 of file common_types.h.

13.92.2.11 uint16

```
typedef uint16_t uint16
```

Definition at line 86 of file common_types.h.

13.92.2.12 uint32

```
typedef uint32_t uint32
```

Definition at line 87 of file common_types.h.

13.92.2.13 uint64

```
typedef uint64_t uint64
```

Definition at line 88 of file common_types.h.

13.92.2.14 uint8

```
typedef uint8_t uint8
```

Definition at line 85 of file common_types.h.

13.92.3 Function Documentation

13.92.3.1 CompileTimeAssert() [1/9]

```
CompileTimeAssert (
            sizeof(uint8)  = =1,
            TypeUint8WrongSize  )
```

13.92.3.2 CompileTimeAssert() [2/9]

```
CompileTimeAssert (
            sizeof(uint16)  = =2,
            TypeUint16WrongSize  )
```

13.92.3.3 CompileTimeAssert() [3/9]

```
CompileTimeAssert (
            sizeof(uint32)  = =4,
            TypeUint32WrongSize  )
```

13.92.3.4 CompileTimeAssert() [4/9]

```
CompileTimeAssert (
            sizeof(uint64)  = =8,
            TypeUint64WrongSize  )
```

13.92.3.5 CompileTimeAssert() [5/9]

```
CompileTimeAssert (
            sizeof(int8)  = =1,
            Typeint8WrongSize  )
```

13.92.3.6 CompileTimeAssert() [6/9]

```
CompileTimeAssert (
            sizeof(int16)  = =2,
            Typeint16WrongSize  )
```

13.92.3.7 CompileTimeAssert() [7/9]

```
CompileTimeAssert (
            sizeof(int32)  = =4,
            Typeint32WrongSize  )
```

13.92.3.8 CompileTimeAssert() [8/9]

```
CompileTimeAssert (
             sizeof(int64)  = =8,
             Typeint64WrongSize  )
```

13.92.3.9 CompileTimeAssert() [9/9]

```
CompileTimeAssert (
             sizeof(cpuaddr) >=sizeof(void *) ,
             TypePtrWrongSize  )
```

13.93 osal/src/os/inc/osapi-os-core.h File Reference

```
#include <stdarg.h>
```

Data Structures

- struct OS_task_prop_t
- struct OS_queue_prop_t
- struct OS_bin_sem_prop_t
- struct OS_count_sem_prop_t
- struct OS_mut_sem_prop_t
- struct OS_time_t
- struct OS_heap_prop_t
- struct OS_FdSet

  An abstract structure capable of holding several OSAL IDs.

Macros

- #define OS_OBJECT_INDEX_MASK 0xFFFF
- #define OS_OBJECT_TYPE_SHIFT 16
- #define OS_OBJECT_TYPE_UNDEFINED 0x00
- #define OS_OBJECT_TYPE_OS_TASK 0x01
- #define OS_OBJECT_TYPE_OS_QUEUE 0x02
- #define OS_OBJECT_TYPE_OS_COUNTSEM 0x03
- #define OS_OBJECT_TYPE_OS_BINSEM 0x04
- #define OS_OBJECT_TYPE_OS_MUTEX 0x05
- #define OS_OBJECT_TYPE_OS_STREAM 0x06
- #define OS_OBJECT_TYPE_OS_DIR 0x07
- #define OS_OBJECT_TYPE_OS_TIMEBASE 0x08
- #define OS_OBJECT_TYPE_OS_TIMECB 0x09
- #define OS_OBJECT_TYPE_OS_MODULE 0x0A
- #define OS_OBJECT_TYPE_OS_FILESYS 0x0B
- #define OS_OBJECT_TYPE_OS_CONSOLE 0x0C
- #define OS_OBJECT_TYPE_USER 0x10
- #define OS_MAX_TASK_PRIORITY 255
- #define OS_SEM_FULL 1
- #define OS_SEM_EMPTY 0
- #define OS_FP_ENABLED 1
- #define OS_ERROR_NAME_LENGTH 35

Typedefs

- typedef char os_err_name_t[OS_ERROR_NAME_LENGTH]
- typedef void osal_task
- typedef void(∗ OS_ArgCallback_t) (uint32 object_id, void ∗arg)

Functions

- typedef osal_task ((∗osal_task_entry)(void))
- void OS_Application_Startup (void)
- void OS_Application_Run (void)
- int32 OS_API_Init (void)

  Initialization of API.
- void OS_IdleLoop (void)

  Background thread implementation - waits forever for events to occur.
- void OS_DeleteAllObjects (void)

  delete all resources created in OSAL.
- void OS_ApplicationShutdown (uint8 flag)

  Initiate orderly shutdown.
- uint32 OS_IdentifyObject (uint32 object_id)

  Obtain the type of an object given an arbitrary object ID.
- int32 OS_ConvertToArrayIndex (uint32 object_id, uint32 ∗ArrayIndex)

  Converts an abstract ID into a number suitable for use as an array index.
- void OS_ForEachObject (uint32 creator_id, OS_ArgCallback_t callback_ptr, void ∗callback_arg)

  call the supplied callback function for all valid object IDs
- int32 OS_TaskCreate (uint32 ∗task_id, const char ∗task_name, osal_task_entry function_pointer, uint32 ∗stack↩ _pointer, uint32 stack_size, uint32 priority, uint32 flags)

  Creates a task and starts running it.
- int32 OS_TaskDelete (uint32 task_id)

  Deletes the specified Task.
- void OS_TaskExit (void)

  Exits the calling task.
- int32 OS_TaskInstallDeleteHandler (osal_task_entry function_pointer)

  Installs a handler for when the task is deleted.
- int32 OS_TaskDelay (uint32 millisecond)

  Delay a task for specified amount of milliseconds.
- int32 OS_TaskSetPriority (uint32 task_id, uint32 new_priority)

  Sets the given task to a new priority.
- int32 OS_TaskRegister (void)

  Registration to be called by new tasks after creation.
- uint32 OS_TaskGetId (void)

  Obtain the task id of the calling task.
- int32 OS_TaskGetIdByName (uint32 ∗task_id, const char ∗task_name)

  Find an existing task ID by name.
- int32 OS_TaskGetInfo (uint32 task_id, OS_task_prop_t ∗task_prop)

  Fill a property object buffer with details regarding the resource.

- int32 OS_QueueCreate (uint32 *queue_id, const char *queue_name, uint32 queue_depth, uint32 data_size, uint32 flags)

  Create a message queue.
- int32 OS_QueueDelete (uint32 queue_id)

  Deletes the specified message queue.
- int32 OS_QueueGet (uint32 queue_id, void *data, uint32 size, uint32 *size_copied, int32 timeout)

  Receive a message on a message queue.
- int32 OS_QueuePut (uint32 queue_id, const void *data, uint32 size, uint32 flags)

  Put a message on a message queue.
- int32 OS_QueueGetIdByName (uint32 *queue_id, const char *queue_name)

  Find an existing queue ID by name.
- int32 OS_QueueGetInfo (uint32 queue_id, OS_queue_prop_t *queue_prop)

  Fill a property object buffer with details regarding the resource.
- int32 OS_BinSemCreate (uint32 *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options)

  Creates a binary semaphore.
- int32 OS_BinSemFlush (uint32 sem_id)

  Unblock all tasks pending on the specified semaphore.
- int32 OS_BinSemGive (uint32 sem_id)

  Increment the semaphore value.
- int32 OS_BinSemTake (uint32 sem_id)

  Decrement the semaphore value.
- int32 OS_BinSemTimedWait (uint32 sem_id, uint32 msecs)

  Decrement the semaphore value with a timeout.
- int32 OS_BinSemDelete (uint32 sem_id)

  Deletes the specified Binary Semaphore.
- int32 OS_BinSemGetIdByName (uint32 *sem_id, const char *sem_name)

  Find an existing semaphore ID by name.
- int32 OS_BinSemGetInfo (uint32 sem_id, OS_bin_sem_prop_t *bin_prop)

  Fill a property object buffer with details regarding the resource.
- int32 OS_CountSemCreate (uint32 *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options)

  Creates a counting semaphore.
- int32 OS_CountSemGive (uint32 sem_id)

  Increment the semaphore value.
- int32 OS_CountSemTake (uint32 sem_id)

  Decrement the semaphore value.
- int32 OS_CountSemTimedWait (uint32 sem_id, uint32 msecs)

  Decrement the semaphore value with timeout.
- int32 OS_CountSemDelete (uint32 sem_id)

  Deletes the specified counting Semaphore.
- int32 OS_CountSemGetIdByName (uint32 *sem_id, const char *sem_name)

  Find an existing semaphore ID by name.
- int32 OS_CountSemGetInfo (uint32 sem_id, OS_count_sem_prop_t *count_prop)

  Fill a property object buffer with details regarding the resource.
- int32 OS_MutSemCreate (uint32 *sem_id, const char *sem_name, uint32 options)

  Creates a mutex semaphore.
- int32 OS_MutSemGive (uint32 sem_id)

  Releases the mutex object referenced by sem_id.

- int32 OS_ShMemSemGive (uint32 Id)
- int32 OS_ShMemAttach (cpuaddr *Address, uint32 Id)
- int32 OS_ShMemGetIdByName (uint32 *ShMemId, const char *SegName)
- int32 OS_HeapGetInfo (OS_heap_prop_t *heap_prop)

  Return current info on the heap.
- int32 OS_GetErrorName (int32 error_num, os_err_name_t *err_name)

  Convert an error number to a string.
- int32 OS_SelectMultiple (OS_FdSet *ReadSet, OS_FdSet *WriteSet, int32 msecs)

  Wait for events across multiple file handles.
- int32 OS_SelectSingle (uint32 objid, uint32 *StateFlags, int32 msecs)

  Wait for events on a single file handle.
- int32 OS_SelectFdZero (OS_FdSet *Set)

  Clear a FdSet structure.
- int32 OS_SelectFdAdd (OS_FdSet *Set, uint32 objid)

  Add an ID to an FdSet structure.
- int32 OS_SelectFdClear (OS_FdSet *Set, uint32 objid)

  Clear an ID from an FdSet structure.
- bool OS_SelectFdIsSet (OS_FdSet *Set, uint32 objid)

  Check if an FdSet structure contains a given ID.
- void OS_printf (const char *string,...) OS_PRINTF(1

  Abstraction for the system printf() call.
- void void OS_printf_disable (void)

  This function disables the output to the console from OS_printf.
- void OS_printf_enable (void)

  This function enables the output to the console through OS_printf.
- void OS_ApplicationExit (int32 Status)

  Exit/Abort the application.

13.93.1 Macro Definition Documentation

13.93.1.1 OS_ERROR_NAME_LENGTH

```
#define OS_ERROR_NAME_LENGTH 35
```

Definition at line 123 of file osapi-os-core.h.

13.93.1.2 OS_FP_ENABLED

```
#define OS_FP_ENABLED 1
```

Definition at line 51 of file osapi-os-core.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateChildTask(), and CFE_ES_CreateObjects().

13.93.1.3 OS_MAX_TASK_PRIORITY

```
#define OS_MAX_TASK_PRIORITY 255
```

Definition at line 44 of file osapi-os-core.h.

13.93.1.4 OS_OBJECT_INDEX_MASK

```
#define OS_OBJECT_INDEX_MASK 0xFFFF
```

Definition at line 25 of file osapi-os-core.h.

13.93.1.5 OS_OBJECT_TYPE_OS_BINSEM

```
#define OS_OBJECT_TYPE_OS_BINSEM 0x04
```

Definition at line 32 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.1.6 OS_OBJECT_TYPE_OS_CONSOLE

```
#define OS_OBJECT_TYPE_OS_CONSOLE 0x0C
```

Definition at line 40 of file osapi-os-core.h.

13.93.1.7 OS_OBJECT_TYPE_OS_COUNTSEM

```
#define OS_OBJECT_TYPE_OS_COUNTSEM 0x03
```

Definition at line 31 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.1.8 OS_OBJECT_TYPE_OS_DIR

```
#define OS_OBJECT_TYPE_OS_DIR 0x07
```

Definition at line 35 of file osapi-os-core.h.

13.93.1.9 OS_OBJECT_TYPE_OS_FILESYS

```
#define OS_OBJECT_TYPE_OS_FILESYS 0x0B
```

Definition at line 39 of file osapi-os-core.h.

13.93.1.10 OS_OBJECT_TYPE_OS_MODULE

```
#define OS_OBJECT_TYPE_OS_MODULE 0x0A
```

Definition at line 38 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.93.1.11 OS_OBJECT_TYPE_OS_MUTEX

```
#define OS_OBJECT_TYPE_OS_MUTEX 0x05
```

Definition at line 33 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.1.12 OS_OBJECT_TYPE_OS_QUEUE

```
#define OS_OBJECT_TYPE_OS_QUEUE 0x02
```

Definition at line 30 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.1.13 OS_OBJECT_TYPE_OS_STREAM

```
#define OS_OBJECT_TYPE_OS_STREAM 0x06
```

Definition at line 34 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.1.14 OS_OBJECT_TYPE_OS_TASK

```
#define OS_OBJECT_TYPE_OS_TASK 0x01
```

Definition at line 29 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.1.15 OS_OBJECT_TYPE_OS_TIMEBASE

```
#define OS_OBJECT_TYPE_OS_TIMEBASE 0x08
```

Definition at line 36 of file osapi-os-core.h.

13.93.1.16 OS_OBJECT_TYPE_OS_TIMECB

```
#define OS_OBJECT_TYPE_OS_TIMECB 0x09
```

Definition at line 37 of file osapi-os-core.h.

Referenced by CFE_ES_CleanupObjectCallback().

13.93.1.17 OS_OBJECT_TYPE_SHIFT

```
#define OS_OBJECT_TYPE_SHIFT 16
```

Definition at line 26 of file osapi-os-core.h.

13.93.1.18 OS_OBJECT_TYPE_UNDEFINED

```
#define OS_OBJECT_TYPE_UNDEFINED 0x00
```

Definition at line 28 of file osapi-os-core.h.

13.93.1.19 OS_OBJECT_TYPE_USER

```
#define OS_OBJECT_TYPE_USER 0x10
```

Definition at line 41 of file osapi-os-core.h.

Referenced by CFE_ES_CountObjectCallback(), CFE_ES_ListResources(), CFE_ES_ListResourcesDebug(), and C↩
FE_ES_ShellCountObjectCallback().

13.93.1.20 OS_SEM_EMPTY

```
#define OS_SEM_EMPTY 0
```

Definition at line 48 of file osapi-os-core.h.

13.93.1.21 OS_SEM_FULL

```
#define OS_SEM_FULL 1
```

Definition at line 47 of file osapi-os-core.h.

13.93.2 Typedef Documentation

13.93.2.1 OS_ArgCallback_t

```
typedef void(* OS_ArgCallback_t) (uint32 object_id, void *arg)
```

Definition at line 136 of file osapi-os-core.h.

13.93.2.2 os_err_name_t

```
typedef char os_err_name_t[OS_ERROR_NAME_LENGTH]
```

Definition at line 124 of file osapi-os-core.h.

13.93.2.3 osal_task

```
typedef void osal_task
```

Definition at line 129 of file osapi-os-core.h.

13.93.3 Function Documentation

13.93.3.1 OS_API_Init()

```
int32 OS_API_Init (
            void  )
```

Initialize the tables that the OS API uses to keep track of information about objects

Returns

on success, or appropriate error code

Referenced by main().

13.93.3.2 OS_Application_Run()

```
void OS_Application_Run (
             void  )
```

13.93.3.3 OS_Application_Startup()

```
void OS_Application_Startup (
             void  )
```

13.93.3.4 OS_ApplicationExit()

```
void OS_ApplicationExit (
             int32 Status )
```

Indicates that the OSAL application should exit and return control to the OS This is intended for e.g. scripted unit testing where the test needs to end without user intervention.

This function does not return. Production code typically should not ever call this.

Note

This exits the entire process including tasks that have been created.

13.93.3.5 OS_ApplicationShutdown()

```
void OS_ApplicationShutdown (
             uint8 flag )
```

Indicates that the OSAL application should perform an orderly shutdown of ALL tasks, clean up all resources, and exit the application.

This allows the task currently blocked in OS_IdleLoop() to wake up, and for that function to return to its caller.

This is preferred over e.g. ApplicationExit() which exits immediately and does not provide for any means to clean up first.

Parameters

flag	set to true to initiate shutdown, false to cancel

Referenced by CFE_PSP_SigintHandler().

13.93.3.6 OS_BinSemCreate()

```
int32 OS_BinSemCreate (
            uint32 * sem_id,
            const char * sem_name,
            uint32 sem_initial_value,
            uint32 options )
```

Creates a binary semaphore with initial value specified by sem_initial_value and name specified by sem_name. sem_id will be returned to the caller

Parameters

out	*sem_id*	will be set to the ID of the newly-created resource
in	*sem_name*	the name of the new resource to create
in	*sem_initial_value*	the initial value of the binary semaphore
in	*options*	Reserved for future use, should be passed as 0.

Returns

on success, or appropriate error code OS_INVALID_POINTER if sen name or sem_id are NULL OS_ERR_N↩ AME_TOO_LONG if the name given is too long OS_ERR_NO_FREE_IDS if all of the semaphore ids are taken OS_ERR_NAME_TAKEN if this is already the name of a binary semaphore OS_SEM_FAILURE if the OS call failed

13.93.3.7 OS_BinSemDelete()

```
int32 OS_BinSemDelete (
            uint32 sem_id )
```

Parameters

in	*sem↩ _id*	The object ID to delete

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid binary semaphore OS_SEM_FAILURE the OS call failed

Referenced by CFE_ES_CleanupObjectCallback().

13.93.3.8 OS_BinSemFlush()

```
int32 OS_BinSemFlush (
            uint32 sem_id )
```

The function unblocks all tasks pending on the specified semaphore. However, this function does not change the state of the semaphore.

Parameters

in	*sem_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a binary semaphore OS_SEM_FAILURE if an unspecified failure occurs

13.93.3.9 OS_BinSemGetIdByName()

```
int32 OS_BinSemGetIdByName (
            uint32 * sem_id,
            const char * sem_name )
```

This function tries to find a binary sem Id given the name of a bin_sem The id is returned through sem_id

Parameters

out	*sem_id*	will be set to the ID of the existing resource
in	*sem_name*	the name of the existing resource to find

Returns

on success, or appropriate error code OS_INVALID_POINTER is semid or sem_name are NULL pointers OS_↩ ERR_NAME_TOO_LONG if the name given is to long to have been stored OS_ERR_NAME_NOT_FOUND if the name was not found in the table

13.93.3.10 OS_BinSemGetInfo()

```
int32 OS_BinSemGetInfo (
            uint32 sem_id,
            OS_bin_sem_prop_t * bin_prop )
```

This function will pass back a pointer to structure that contains all of the relevant info(name and creator) about the specified binary semaphore.

Parameters

in	*sem_id*	The object ID to operate on
out	*bin_prop*	The property object buffer to fill

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid semaphore OS_I↩ NVALID_POINTER if the bin_prop pointer is null OS_SUCCESS if success

13.93.3.11 OS_BinSemGive()

```
int32 OS_BinSemGive (
            uint32 sem_id )
```

The function unlocks the semaphore referenced by sem_id by performing a semaphore unlock operation on that semaphore. If the semaphore value resulting from this operation is positive, then no threads were blocked waiting for the semaphore to become unlocked; the semaphore value is simply incremented for this semaphore.

Parameters

in	*sem↩ _id*	The object ID to operate on

Returns

on success, or appropriate error code OS_SEM_FAILURE the semaphore was not previously initialized or is not in the array of semaphores defined by the system OS_ERR_INVALID_ID if the id passed in is not a binary semaphore

13.93.3.12 OS_BinSemTake()

```
int32 OS_BinSemTake (
            uint32 sem_id )
```

The locks the semaphore referenced by sem_id by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call until it either locks the semaphore or the call is interrupted.

Parameters

in	*sem_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_ERR_INVALID_ID the Id passed in is not a valid binary semaphore OS_SEM_FAILURE if the OS call failed

13.93.3.13 OS_BinSemTimedWait()

```
int32 OS_BinSemTimedWait (
            uint32 sem_id,
            uint32 msecs )
```

The function locks the semaphore referenced by sem_id. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore, this wait shall be terminated when the specified timeout, msecs, expires.

Parameters

in	*sem_id*	The object ID to operate on
in	*msecs*	The maximum amount of time to block, in milliseconds

Returns

on success, or appropriate error code OS_SEM_TIMEOUT if semaphore was not relinquished in time OS_SEM↩ _FAILURE the semaphore was not previously initialized or is not in the array of semaphores defined by the system OS_ERR_INVALID_ID if the ID passed in is not a valid semaphore ID

13.93.3.14 OS_ConvertToArrayIndex()

```
int32 OS_ConvertToArrayIndex (
            uint32 object_id,
            uint32 * ArrayIndex )
```

This will return a unique zero-based integer number in the range of [0,MAX) for any valid object ID. This may be used by application code as an array index for indexing into local tables.

Note

This does NOT verify the validity of the ID, that is left to the caller. This is only the conversion logic.

Parameters

in	*object_id*	The object ID to operate on
out	*∗ArrayIndex*	The Index to return

Returns

on success, or appropriate error code

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanupTaskResources(), CFE_ES_CreateChildTask(), CFE_ES_↩ CreateObjects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), CFE_ES_GetAppIDInternal(), CFE_ES_Get↩ AppInfoInternal(), CFE_ES_GetTaskInfo(), CFE_ES_IncrementTaskCounter(), CFE_ES_RunLoop(), CFE_SB_Finish↩ SendEvent(), and CFE_SB_RequestToSendEvent().

13.93.3.15 OS_CountSemCreate()

```
int32 OS_CountSemCreate (
            uint32 * sem_id,
            const char * sem_name,
            uint32 sem_initial_value,
            uint32 options )
```

Creates a counting semaphore with initial value specified by sem_initial_value and name specified by sem_name. sem_id will be returned to the caller

Parameters

out	*sem_id*	will be set to the ID of the newly-created resource
in	*sem_name*	the name of the new resource to create
in	*sem_initial_value*	the initial value of the counting semaphore
in	*options*	Reserved for future use, should be passed as 0.

Returns

on success, or appropriate error code OS_INVALID_POINTER if sen name or sem_id are NULL OS_ERR_N↩ AME_TOO_LONG if the name given is too long OS_ERR_NO_FREE_IDS if all of the semaphore ids are taken OS_ERR_NAME_TAKEN if this is already the name of a counting semaphore OS_SEM_FAILURE if the OS call failed OS_INVALID_SEM_VALUE if the semaphore value is too high

13.93.3.16 OS_CountSemDelete()

```
int32 OS_CountSemDelete (
            uint32 sem_id )
```

Parameters

in	*sem↩ _id*	The object ID to delete

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid counting semaphore OS_SEM_FAILURE the OS call failed

Referenced by CFE_ES_CleanupObjectCallback().

13.93.3.17 OS_CountSemGetIdByName()

```
int32 OS_CountSemGetIdByName (
            uint32 * sem_id,
            const char * sem_name )
```

This function tries to find a counting sem Id given the name of a count_sem The id is returned through sem_id

Parameters

out	*sem_id*	will be set to the ID of the existing resource
in	*sem_name*	the name of the existing resource to find

Returns

on success, or appropriate error code OS_INVALID_POINTER is semid or sem_name are NULL pointers OS_↩ ERR_NAME_TOO_LONG if the name given is to long to have been stored OS_ERR_NAME_NOT_FOUND if the name was not found in the table

13.93.3.18 OS_CountSemGetInfo()

```
int32 OS_CountSemGetInfo (
            uint32 sem_id,
            OS_count_sem_prop_t * count_prop )
```

This function will pass back a pointer to structure that contains all of the relevant info(name and creator) about the specified counting semaphore.

Parameters

in	*sem_id*	The object ID to operate on
out	*count_prop*	The property object buffer to fill

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid semaphore OS_I↩ NVALID_POINTER if the count_prop pointer is null

13.93.3.19 OS_CountSemGive()

```
int32 OS_CountSemGive (
            uint32 sem_id )
```

The function unlocks the semaphore referenced by sem_id by performing a semaphore unlock operation on that semaphore. If the semaphore value resulting from this operation is positive, then no threads were blocked waiting for the semaphore to become unlocked; the semaphore value is simply incremented for this semaphore.

Parameters

in	*sem_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_SEM_FAILURE the semaphore was not previously initialized or is not in the array of semaphores defined by the system OS_ERR_INVALID_ID if the id passed in is not a counting semaphore

13.93.3.20 OS_CountSemTake()

```
int32 OS_CountSemTake (
            uint32 sem_id )
```

The locks the semaphore referenced by sem_id by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call until it either locks the semaphore or the call is interrupted.

Parameters

in	*sem_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_ERR_INVALID_ID the Id passed in is not a valid counting semaphore OS_SEM_FAILURE if the OS call failed

13.93.3.21 OS_CountSemTimedWait()

```
int32 OS_CountSemTimedWait (
            uint32 sem_id,
            uint32 msecs )
```

The function locks the semaphore referenced by sem_id. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore, this wait shall be terminated when the specified timeout, msecs, expires.

Parameters

in	*sem_id*	The object ID to operate on
in	*msecs*	The maximum amount of time to block, in milliseconds

Returns

on success, or appropriate error code OS_SEM_TIMEOUT if semaphore was not relinquished in time OS_SEM_FAILURE the semaphore was not previously initialized or is not in the array of semaphores defined by the system OS_ERR_INVALID_ID if the ID passed in is not a valid semaphore ID

13.93.3.22 OS_DeleteAllObjects()

```
void OS_DeleteAllObjects (
            void  )
```

provides a means to clean up all resources allocated by this instance of OSAL. It would typically be used during an orderly shutdown but may also be helpful for testing purposes.

Referenced by main().

13.93.3.23 OS_ExcAttachHandler()

```
int32 OS_ExcAttachHandler (
            uint32 ExceptionNumber,
            void(*)(uint32, const void *, uint32) ExceptionHandler,
            int32 parameter )
```

13.93.3.24 OS_ExcDisable()

```
int32 OS_ExcDisable (
            int32 ExceptionNumber )
```

13.93.3.25 OS_ExcEnable()

```
int32 OS_ExcEnable (
            int32 ExceptionNumber )
```

13.93.3.26 OS_ForEachObject()

```
void OS_ForEachObject (
            uint32 creator_id,
            OS_ArgCallback_t callback_ptr,
            void * callback_arg )
```

Loops through all defined OSAL objects and calls callback_ptr on each one If creator_id is nonzero then only objects with matching creator id are processed.

Referenced by CFE_ES_CleanupTaskResources(), CFE_ES_ListResources(), and CFE_ES_ListResourcesDebug().

13.93.3.27 OS_FPUExcAttachHandler()

```
int32 OS_FPUExcAttachHandler (
            uint32 ExceptionNumber,
            osal_task_entry ExceptionHandler,
            int32 parameter )
```

Parameters

in	*ExceptionNumber*	The exception number to attach to
in	*ExceptionHandler*	Pointer to handler function
in	*parameter*	Argument to pass to handler

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.28 OS_FPUExcDisable()

```
int32 OS_FPUExcDisable (
            int32 ExceptionNumber )
```

Parameters

in	*ExceptionNumber*	The exception number to disable

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.29 OS_FPUExcEnable()

```
int32 OS_FPUExcEnable (
            int32 ExceptionNumber )
```

Parameters

in	*ExceptionNumber*	The exception number to enable

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.30 OS_FPUExcGetMask()

```
int32 OS_FPUExcGetMask (
            uint32 * mask )
```

This function gets the FPU exception mask

Note

The exception environment is local to each task Therefore this must be called for each task that that wants to do floating point and catch exceptions.

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.31 OS_FPUExcSetMask()

```
int32 OS_FPUExcSetMask (
            uint32 mask )
```

This function sets the FPU exception mask

Note

The exception environment is local to each task Therefore this must be called for each task that that wants to do floating point and catch exceptions.

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.32 OS_GetErrorName()

```
int32 OS_GetErrorName (
            int32 error_num,
            os_err_name_t * err_name )
```

Parameters

in	*error_num*	Error number to convert
out	*err_name*	Buffer to store error string

Returns

on success, or appropriate error code

13.93.3.33 OS_GetLocalTime()

```
int32 OS_GetLocalTime (
            OS_time_t * time_struct )
```

This function gets the local time of the machine its on

Parameters

out	*time_struct*	An OS_time_t that will be set to the current time

Returns

on success, or appropriate error code

Referenced by CFE_PSP_Get_Timebase(), and CFE_PSP_GetTime().

13.93.3.34 OS_HeapGetInfo()

```
int32 OS_HeapGetInfo (
            OS_heap_prop_t * heap_prop )
```

Parameters

out	*heap_prop*	Storage buffer for heap info

Returns

on success, or appropriate error code

Referenced by CFE_ES_HousekeepingCmd().

13.93.3.35 OS_IdentifyObject()

```
uint32 OS_IdentifyObject (
            uint32 object_id )
```

Given an arbitrary object ID, get the type of the object

Parameters

in	*object_id*	The object ID to operate on

Returns

type of object that the ID represents OS_OBJECT_TYPE_OS_TASK for tasks OS_OBJECT_TYPE_OS_QUEUE for queues, etc.

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_CountObjectCallback(), and CFE_ES_ShellCount↩

ObjectCallback().

13.93.3.36 OS_IdleLoop()

```
void OS_IdleLoop (
            void  )
```

This should be called from the BSP main routine / initial thread after all other board / application initialization has taken place and all other tasks are running.

Typically just waits forever until "OS_shutdown" flag becomes true.

Referenced by main().

13.93.3.37 OS_IntAck()

```
int32 OS_IntAck (
            int32 InterruptNumber )
```

13.93.3.38 OS_IntAttachHandler()

```
int32 OS_IntAttachHandler (
            uint32 InterruptNumber,
            osal_task_entry InterruptHandler,
            int32 parameter )
```

The call associates a specified C routine to a specified interrupt number. Upon occurring of the InterruptNumber, the InerruptHandler routine will be called and passed the parameter.

Parameters

in	*InterruptNumber*	The Interrupt Number that will cause the start of the ISR
in	*InterruptHandler*	The ISR associated with this interrupt
in	*parameter*	Argument that is passed to the ISR

Returns

on success or appropriate error code OS_INVALID_POINTER if the Interrupt handler pointer is NULL OS_ERR↩ _NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.39 OS_IntDisable()

```
int32 OS_IntDisable (
            int32 Level )
```

Parameters

in	*Level*	the interrupts to disable

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.40 OS_IntEnable()

```
int32 OS_IntEnable (
            int32 Level )
```

Parameters

in	*Level*	the interrupts to enable

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.41 OS_IntGetMask()

```
int32 OS_IntGetMask (
            uint32 * mask )
```

Note

The interrupt bits are architecture-specific.

Parameters

out	*mask*	The register value will be stored to this location

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

13.93.3.42 OS_IntLock()

```
int32 OS_IntLock (
            void  )
```

Returns

key value to be passed to OS_IntUnlock() to restore interrupts OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

Referenced by CFE_ES_PerfLogAdd().

13.93.3.43 OS_IntSetMask()

```
int32 OS_IntSetMask (
            uint32 mask )
```

Note

The interrupt bits are architecture-specific.

Parameters

in	*mask*	The value to set in the register

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this func-

tion

13.93.3.44 OS_IntUnlock()

```
int32 OS_IntUnlock (
            int32 IntLevel )
```

Parameters

in	*IntLevel*	value from previous call to OS_IntLock()

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED on platforms that do not support this function

Referenced by CFE_ES_PerfLogAdd().

13.93.3.45 OS_Milli2Ticks()

```
int32 OS_Milli2Ticks (
            uint32 milli_seconds )
```

This function accepts a time interval in milliseconds and returns the tick equivalent. If the result is not an exact number of system ticks, the result will be rounded up to the nearest tick.

Parameters

in	*milli_seconds*	the number of milliseconds

Returns

number of ticks

13.93.3.46 OS_MutSemCreate()

```
int32 OS_MutSemCreate (
            uint32 * sem_id,
```

```
            const char * sem_name,
            uint32 options )
```

Mutex semaphores are always created in the unlocked (full) state.

Parameters

out	*sem_id*	will be set to the ID of the newly-created resource
in	*sem_name*	the name of the new resource to create
in	*options*	reserved for future use. Should be passed as 0.

Returns

on success, or appropriate error code OS_INVALID_POINTER if sem_id or sem_name are NULL OS_ERR_N↩ AME_TOO_LONG if the sem_name is too long to be stored OS_ERR_NO_FREE_IDS if there are no more free mutex Ids OS_ERR_NAME_TAKEN if there is already a mutex with the same name OS_SEM_FAILURE if the OS call failed

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_CreateCDSPool(), CFE_ES_Main(), CFE_ES_PoolCreateEx(), C↩ FE_ES_RebuildCDSPool(), CFE_EVS_EarlyInit(), and CFE_SB_EarlyInit().

13.93.3.47 OS_MutSemDelete()

```
int32 OS_MutSemDelete (
            uint32 sem_id )
```

Parameters

in	*sem↩ _id*	The object ID to delete

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid mutex OS_SEM_↩ FAILURE if the OS call failed

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_CreateCDSPool(), and CFE_ES_RebuildCDSPool().

13.93.3.48 OS_MutSemGetIdByName()

```
int32 OS_MutSemGetIdByName (
            uint32 * sem_id,
            const char * sem_name )
```

This function tries to find a mutex sem Id given the name of a mut_sem The id is returned through sem_id

Parameters

out	*sem_id*	will be set to the ID of the existing resource
in	*sem_name*	the name of the existing resource to find

Returns

on success, or appropriate error code OS_INVALID_POINTER is semid or sem_name are NULL pointers OS_↩ ERR_NAME_TOO_LONG if the name given is to long to have been stored OS_ERR_NAME_NOT_FOUND if the name was not found in the table

13.93.3.49 OS_MutSemGetInfo()

```
int32 OS_MutSemGetInfo (
            uint32 sem_id,
            OS_mut_sem_prop_t * mut_prop )
```

This function will pass back a pointer to structure that contains all of the relevant info(name and creator) about the specified mutex semaphore.

Parameters

in	*sem_id*	The object ID to operate on
out	*mut_prop*	The property object buffer to fill

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid semaphore OS_I↩ NVALID_POINTER if the mut_prop pointer is null

13.93.3.50 OS_MutSemGive()

```
int32 OS_MutSemGive (
            uint32 sem_id )
```

If there are threads blocked on the mutex object referenced by mutex when this function is called, resulting in the mutex becoming available, the scheduling policy shall determine which thread shall acquire the mutex.

Parameters

in	*sem_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid mutex OS_SEM_FAILURE if an unspecified error occurs

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_PutCDSBlock(), CFE_ES_PutPoolBuf(), CFE_ES_RebuildCDSPool(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UnlockSharedData(), CFE_EVS_SetLogModeCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_UnlockSharedData(), EVS_AddLog(), and EVS_ClearLog().

13.93.3.51 OS_MutSemTake()

```
int32 OS_MutSemTake (
            uint32 sem_id )
```

If the mutex is already locked, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by mutex in the locked state with the calling thread as its owner.

Parameters

in	*sem_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_SEM_FAILURE if the semaphore was not previously initialized or is not in the array of semaphores defined by the system OS_ERR_INVALID_ID the id passed in is not a valid mutex

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_CreateCDSPool(), CFE_ES_GetCDSBlock(), CFE_ES_GetPoolBuf(), CFE_ES_GetPoolBufInfo(), CFE_ES_LockCDSRegistry(), CFE_ES_LockSharedData(), CFE_ES_PutCDSBlock(), CFE_ES_PutPoolBuf(), CFE_ES_RebuildCDSPool(), CFE_EVS_SetLogModeCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_LockSharedData(), EVS_AddLog(), and EVS_ClearLog().

13.93.3.52 OS_printf()

```
void OS_printf (
            const char * string,
             ...  )
```

This function abstracts out the printf type statements. This is useful for using OS- specific thats that will allow non-polled print statements for the real time systems.

Operates in a manner similar to the printf() call defined by the standard C library. This abstraction may implement additional buffering, if necessary, to improve the real-time performance of the call.

The output of this routine also may be dynamically enabled or disabled by the OS_printf_enable() and OS_printf_↩ disable() calls, respectively.

Parameters

in	*string*	Format string, followed by additional arguments

Referenced by CFE_ES_CleanUpApp(), CFE_ES_ListResourcesDebug(), CFE_ES_ScanAppTable(), CFE_ES_↩ SetupResetVariables(), CFE_ES_SysLogWrite_Unsync(), CFE_ES_WriteToSysLog(), CFE_PSP_AttachExceptions(), CFE_PSP_InitCDS(), CFE_PSP_InitProcessorReservedMemory(), CFE_PSP_InitResetArea(), CFE_PSP_InitUser↩ ReservedArea(), CFE_PSP_Panic(), CFE_PSP_Restart(), CFE_PSP_SetupLocal1Hz(), EVS_OutputPort1(), EVS_↩ OutputPort2(), EVS_OutputPort3(), EVS_OutputPort4(), and main().

13.93.3.53 OS_printf_disable()

```
void void OS_printf_disable (
            void  )
```

13.93.3.54 OS_printf_enable()

```
void OS_printf_enable (
            void  )
```

13.93.3.55 OS_QueueCreate()

```
int32 OS_QueueCreate (
            uint32 * queue_id,
            const char * queue_name,
            uint32 queue_depth,
            uint32 data_size,
            uint32 flags )
```

Parameters

out	*queue_id*	will be set to the ID of the newly-created resource
in	*queue_name*	the name of the new resource to create
in	*queue_depth*	the maximum depth of the queue
in	*data_size*	the size of each entry in the queue
in	*flags*	options for the queue (reserved for future use, pass as 0)

Returns

on success, or appropriate error code OS_INVALID_POINTER if a pointer passed in is NULL OS_ERR_NAM↩ E_TOO_LONG if the name passed in is too long OS_ERR_NO_FREE_IDS if there are already the max queues created OS_ERR_NAME_TAKEN if the name is already being used on another queue OS_ERROR if the OS create call fails

Referenced by CFE_SB_CreatePipe().

13.93.3.56 OS_QueueDelete()

```
int32 OS_QueueDelete (
            uint32 queue_id )
```

Note

If There are messages on the queue, they will be lost and any subsequent calls to QueueGet or QueuePut to this queue will result in errors

Parameters

in	*queue↩ _id*	The object ID to delete

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in does not exist OS_ERROR if the OS call to delete the queue fails

Referenced by CFE_ES_CleanupObjectCallback(), and CFE_SB_DeletePipeFull().

13.93.3.57 OS_QueueGet()

```
int32 OS_QueueGet (
            uint32 queue_id,
            void * data,
            uint32 size,
            uint32 * size_copied,
            int32 timeout )
```

If a message is pending, it is returned immediately. Otherwise the calling task will block until a message arrives or the timeout expires.

Parameters

in	*queue_id*	The object ID to operate on
out	*data*	The buffer to store the received message
in	*size*	The size of the data buffer
out	*size_copied*	Set to the actual size of the message
in	*timeout*	The maximum amount of time to block, or OS_PEND to wait forever

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the given ID does not exist OS_ERR_INVALID_↩ POINTER if a pointer passed in is NULL OS_QUEUE_EMPTY if the Queue has no messages on it to be recieved OS_QUEUE_TIMEOUT if the timeout was OS_PEND and the time expired OS_QUEUE_INVALID_SIZE if the size copied from the queue was not correct

Referenced by CFE_SB_ReadQueue().

13.93.3.58 OS_QueueGetIdByName()

```
int32 OS_QueueGetIdByName (
            uint32 * queue_id,
            const char * queue_name )
```

This function tries to find a queue Id given the name of the queue. The id of the queue is passed back in queue_id.

Parameters

out	*queue_id*	will be set to the ID of the existing resource
in	*queue_name*	the name of the existing resource to find

Returns

on success, or appropriate error code OS_INVALID_POINTER if the name or id pointers are NULL OS_ERR_↩ NAME_TOO_LONG the name passed in is too long OS_ERR_NAME_NOT_FOUND the name was not found in the table

Referenced by CFE_SB_GetPipeIdByName().

13.93.3.59 OS_QueueGetInfo()

```
int32 OS_QueueGetInfo (
            uint32 queue_id,
            OS_queue_prop_t * queue_prop )
```

This function will pass back a pointer to structure that contains all of the relevant info (name and creator) about the specified queue.

Parameters

in	*queue_id*	The object ID to operate on
out	*queue_prop*	The property object buffer to fill

Returns

on success, or appropriate error code OS_INVALID_POINTER if queue_prop is NULL OS_ERR_INVALID_ID if the ID given is not a valid queue OS_SUCCESS if the info was copied over correctly

Referenced by CFE_SB_GetPipeName().

13.93.3.60 OS_QueuePut()

```
int32 OS_QueuePut (
            uint32 queue_id,
            const void * data,
            uint32 size,
            uint32 flags )
```

Parameters

in	*queue_id*	The object ID to operate on
in	*data*	The buffer containing the message to put
in	*size*	The size of the data buffer
in	*flags*	Currently reserved/unused, should be passed as 0

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the queue id passed in is not a valid queue OS↩ _INVALID_POINTER if the data pointer is NULL OS_QUEUE_FULL if the queue cannot accept another message OS_ERROR if the OS call returns an error

Referenced by CFE_SB_SendMsgFull().

13.93.3.61 OS_SelectFdAdd()

```
int32 OS_SelectFdAdd (
            OS_FdSet * Set,
            uint32 objid )
```

After this call the set will contain the given OSAL ID

Returns

on success, or appropriate error code

13.93.3.62 OS_SelectFdClear()

```
int32 OS_SelectFdClear (
            OS_FdSet * Set,
            uint32 objid )
```

After this call the set will no longer contain the given OSAL ID

Returns

on success, or appropriate error code

13.93.3.63 OS_SelectFdIsSet()

```
bool OS_SelectFdIsSet (
            OS_FdSet * Set,
            uint32 objid )
```

13.93.3.64 OS_SelectFdZero()

```
int32 OS_SelectFdZero (
            OS_FdSet * Set )
```

After this call the set will contain no OSAL IDs

Returns

on success, or appropriate error code

13.93.3.65 OS_SelectMultiple()

```
int32 OS_SelectMultiple (
            OS_FdSet * ReadSet,
            OS_FdSet * WriteSet,
            int32 msecs )
```

Wait for any of the given sets of IDs to be become readable or writable

This function will block until any of the following occurs:

- At least one OSAL ID in the ReadSet is readable
- At least one OSAL ID in the WriteSet is writable
- The timeout has elapsed

The sets are input/output parameters. On entry, these indicate the file handle(s) to wait for. On exit, these are set to the actual file handle(s) that have activity.

If the timeout occurs this returns an error code and all output sets should be empty.

Note

This does not lock or otherwise protect the file handles in the given sets. If a filehandle supplied via one of the FdSet arguments is closed or modified by another while this function is in progress, the results are undefined. Because of this limitation, it is recommended to use OS_SelectSingle() whenever possible.

Returns

on success, or appropriate error code

13.93.3.66 OS_SelectSingle()

```
int32 OS_SelectSingle (
            uint32 objid,
            uint32 * StateFlags,
            int32 msecs )
```

Wait for a single OSAL filehandle to change state

This function can be used to wait for a single OSAL stream ID to become readable or writable. On entry, the "State↩ Flags" parameter should be set to the desired state (readble or writable) and upon return the flags will be set to the state actually detected.

As this operates on a single ID, the filehandle is protected during this call, such that another thread accessing the same handle will return an error. However, it is important to note that once the call returns then other threads may then also read/write and affect the state before the current thread can service it.

To mitigate this risk the application may prefer to use the OS_TimedRead/OS_TimedWrite calls.

Returns

on success, or appropriate error code

13.93.3.67 OS_SetLocalTime()

```
int32 OS_SetLocalTime (
            OS_time_t * time_struct )
```

This function sets the local time of the machine its on

Parameters

in	*time_struct*	An OS_time_t containing the current time

Returns

on success, or appropriate error code

13.93.3.68 OS_ShMemAttach()

```
int32 OS_ShMemAttach (
            cpuaddr * Address,
            uint32 Id )
```

13.93.3.69 OS_ShMemCreate()

```
int32 OS_ShMemCreate (
            uint32 * Id,
            uint32 NBytes,
            const char * SegName )
```

13.93.3.70 OS_ShMemGetIdByName()

```
int32 OS_ShMemGetIdByName (
            uint32 * ShMemId,
            const char * SegName )
```

13.93.3.71 OS_ShMemInit()

```
int32 OS_ShMemInit (
            void  )
```

13.93.3.72 OS_ShMemSemGive()

```
int32 OS_ShMemSemGive (
            uint32 Id )
```

13.93.3.73 OS_ShMemSemTake()

```
int32 OS_ShMemSemTake (
            uint32 Id )
```

13.93.3.74 OS_TaskCreate()

```
int32 OS_TaskCreate (
            uint32 * task_id,
            const char * task_name,
            osal_task_entry function_pointer,
            uint32 * stack_pointer,
            uint32 stack_size,
            uint32 priority,
            uint32 flags )
```

Parameters

out	*task_id*	will be set to the ID of the newly-created resource
in	*task_name*	the name of the new resource to create
in	*function_pointer*	the entry point of the new task
in	*stack_pointer*	pointer to the stack for the task, or NULL to allocate a stack from the system memory heap
in	*stack_size*	the size of the stack, or 0 to use a default stack size.
in	*priority*	initial priority of the new task
in	*flags*	initial options for the new task

Returns

on success, or appropriate error code OS_INVALID_POINTER if any of the necessary pointers are NULL OS↩ _ERR_NAME_TOO_LONG if the name of the task is too long to be copied OS_ERR_INVALID_PRIORITY if the priority is bad OS_ERR_NO_FREE_IDS if there can be no more tasks created OS_ERR_NAME_TAKEN if the name specified is already used by a task OS_ERROR if an unspecified/other error occurs

Referenced by CFE_ES_AppCreate(), CFE_ES_CreateChildTask(), and CFE_ES_CreateObjects().

13.93.3.75 OS_TaskDelay()

```
int32 OS_TaskDelay (
            uint32 millisecond )
```

Parameters

in	*millisecond*	Amount of time to delay

Returns

on success, or appropriate error code OS_ERROR if sleep fails or millisecond = 0 OS_SUCCESS if success

Referenced by CFE_ES_CleanUpApp(), CFE_ES_CreateObjects(), CFE_ES_ExitApp(), CFE_ES_InitializeFile↩ Systems(), CFE_ES_Main(), CFE_ES_MainTaskSyncDelay(), CFE_ES_PerfLogDump(), CFE_ES_SetupReset↩ Variables(), CFE_ES_ShellOutputCommand(), and CFE_ES_WaitForSystemState().

13.93.3.76 OS_TaskDelete()

```
int32 OS_TaskDelete (
            uint32 task_id )
```

The task will be removed from the local tables. and the OS will be configured to stop executing the task at the next opportunity.

Parameters

in	*task_id*	The object ID to operate on

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the ID given to it is invalid OS_ERROR if the OS delete call fails

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_CleanupTaskResources(), and CFE_ES_DeleteChildTask().

13.93.3.77 OS_TaskExit()

```
void OS_TaskExit (
             void  )
```

The calling thread is terminated. This function does not return.

Referenced by CFE_ES_ExitApp(), and CFE_ES_ExitChildTask().

13.93.3.78 OS_TaskGetId()

```
uint32 OS_TaskGetId (
             void  )
```

This function returns the task id of the calling task

Returns

ID, or zero if the operation failed (zero is never a valid task ID)

Referenced by CFE_ES_CreateChildTask(), CFE_ES_ExitChildTask(), CFE_ES_GetAppIDInternal(), CFE_ES_IncrementTaskCounter(), CFE_SB_CreatePipe(), CFE_SB_DeletePipeFull(), CFE_SB_GetLastSenderId(), CFE_SB_GetPipeIdByName(), CFE_SB_GetPipeOpts(), CFE_SB_RcvMsg(), CFE_SB_SendMsgFull(), CFE_SB_SetPipeOpts(), CFE_SB_SubscribeFull(), and CFE_SB_UnsubscribeFull().

13.93.3.79 OS_TaskGetIdByName()

```
int32 OS_TaskGetIdByName (
            uint32 * task_id,
            const char * task_name )
```

This function tries to find a task Id given the name of a task

Parameters

out	*task_id*	will be set to the ID of the existing resource
in	*task_name*	the name of the existing resource to find

Returns

on success, or appropriate error code OS_INVALID_POINTER if the pointers passed in are NULL OS_ERR_↩ NAME_TOO_LONG if th ename to found is too long to begin with OS_ERR_NAME_NOT_FOUND if the name wasn't found in the table

13.93.3.80 OS_TaskGetInfo()

```
int32 OS_TaskGetInfo (
            uint32 task_id,
            OS_task_prop_t * task_prop )
```

This function will pass back a pointer to structure that contains all of the relevant info (creator, stack size, priority, name) about the specified task.

Parameters

in	*task_id*	The object ID to operate on
out	*task_prop*	The property object buffer to fill

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the ID passed to it is invalid OS_INVALID_POI↩ NTER if the task_prop pointer is NULL

Referenced by CFE_ES_ProcessCoreException().

13.93.3.81 OS_TaskInstallDeleteHandler()

```
int32 OS_TaskInstallDeleteHandler (
            osal_task_entry function_pointer )
```

Parameters

in	*function_pointer*	function to be called when task exits

Returns

on success, or appropriate error code

13.93.3.82 OS_TaskRegister()

```
int32 OS_TaskRegister (
            void  )
```

Obsolete function retained for compatibility purposes. Does Nothing in the current implementation.

Returns

(always)

Referenced by CFE_ES_RegisterApp(), and CFE_ES_RegisterChildTask().

13.93.3.83 OS_TaskSetPriority()

```
int32 OS_TaskSetPriority (
            uint32 task_id,
            uint32 new_priority )
```

Parameters

in	*task_id*	The object ID to operate on
in	*new_priority*	Set the new priority

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the ID passed to it is invalid OS_ERR_INVALID_PRIORITY if the priority is greater than the max allowed OS_ERROR if the OS call to change the priority fails

13.93.3.84 OS_Tick2Micros()

```
int32 OS_Tick2Micros (
            void  )
```

This function returns the duration of a system tick in micro seconds

Note

care is taken to ensure this does not return "0" since it is often used as the divisor in mathematical operations

Returns

of a system tick in microseconds

13.93.3.85 osal_task()

```
typedef osal_task (
            (*)(void) osal_task_entry )
```

13.94 osal/src/os/inc/osapi-os-filesys.h File Reference

Data Structures

- struct OS_VolumeInfo_t
- struct os_fsinfo_t
- struct OS_file_prop_t
- struct os_fstat_t
- struct os_dirent_t

Macros

- #define OS_READ_ONLY 0
- #define OS_WRITE_ONLY 1
- #define OS_READ_WRITE 2
- #define OS_SEEK_SET 0
- #define OS_SEEK_CUR 1
- #define OS_SEEK_END 2
- #define OS_CHK_ONLY 0
- #define OS_REPAIR 1
- #define FS_BASED 0
- #define RAM_DISK 1
- #define EEPROM_DISK 2
- #define ATA_DISK 3
- #define NUM_TABLE_ENTRIES 14
- #define OS_FS_DEV_NAME_LEN 32
- #define OS_FS_PHYS_NAME_LEN 64
- #define OS_FS_VOL_NAME_LEN 32
- #define OS_FS_ERR_PATH_TOO_LONG (-103)
- #define OS_FS_ERR_NAME_TOO_LONG (-104)
- #define OS_FS_ERR_DRIVE_NOT_CREATED (-106)
- #define OS_FS_ERR_DEVICE_NOT_FREE (-107)
- #define OS_FS_ERR_PATH_INVALID (-108)
- #define OS_FS_SUCCESS OS_SUCCESS
- #define OS_FS_ERROR OS_ERROR
- #define OS_FS_ERR_INVALID_POINTER OS_INVALID_POINTER
- #define OS_FS_ERR_NO_FREE_FDS OS_ERR_NO_FREE_IDS
- #define OS_FS_ERR_INVALID_FD OS_ERR_INVALID_ID
- #define OS_FS_UNIMPLEMENTED OS_ERR_NOT_IMPLEMENTED
- #define OS_FILESTAT_MODE(x) ((x).FileModeBits)
- #define OS_FILESTAT_ISDIR(x) ((x).FileModeBits & OS_FILESTAT_MODE_DIR)
- #define OS_FILESTAT_EXEC(x) ((x).FileModeBits & OS_FILESTAT_MODE_EXEC)
- #define OS_FILESTAT_WRITE(x) ((x).FileModeBits & OS_FILESTAT_MODE_WRITE)
- #define OS_FILESTAT_READ(x) ((x).FileModeBits & OS_FILESTAT_MODE_READ)
- #define OS_FILESTAT_SIZE(x) ((x).FileSize)
- #define OS_FILESTAT_TIME(x) ((x).FileTime)
- #define OS_DIRENTRY_NAME(x) ((x).FileName)

Typedefs

- typedef os_err_name_t os_fs_err_name_t
- typedef void * os_dirp_t
- typedef int32 os_fshealth_t
- typedef OS_file_prop_t OS_FDTableEntry

Enumerations

- enum { OS_FILESTAT_MODE_EXEC = 0x00001, OS_FILESTAT_MODE_WRITE = 0x00002, OS_FILESTAT↩ _MODE_READ = 0x00004, OS_FILESTAT_MODE_DIR = 0x10000 }

Functions

- int32 OS_creat (const char *path, int32 access)

  Creates a file specified by path.
- int32 OS_open (const char *path, int32 access, uint32 mode)

  Opens a file.
- int32 OS_close (uint32 filedes)

  Closes an open file handle.
- int32 OS_read (uint32 filedes, void *buffer, uint32 nbytes)

  Read from a file handle.
- int32 OS_write (uint32 filedes, const void *buffer, uint32 nbytes)

  Write to a file handle.
- int32 OS_TimedRead (uint32 filedes, void *buffer, uint32 nbytes, int32 timeout)

  File/Stream input read with a timeout.
- int32 OS_TimedWrite (uint32 filedes, const void *buffer, uint32 nbytes, int32 timeout)

  File/Stream output write with a timeout.
- int32 OS_chmod (const char *path, uint32 access)

  Changes the permissions of a file.
- int32 OS_stat (const char *path, os_fstat_t *filestats)

  Obtain information about a file or directory.
- int32 OS_lseek (uint32 filedes, int32 offset, uint32 whence)

  Seeks to the specified position of an open file.
- int32 OS_remove (const char *path)

  Removes a file from the file system.
- int32 OS_rename (const char *old_filename, const char *new_filename)

  Renames a file.
- int32 OS_cp (const char *src, const char *dest)

  Copies a single file from src to dest.
- int32 OS_mv (const char *src, const char *dest)

  Move a single file from src to dest.
- int32 OS_FDGetInfo (uint32 filedes, OS_file_prop_t *fd_prop)

  Obtain information about an open file.
- int32 OS_FileOpenCheck (const char *Filename)

  Checks to see if a file is open.
- int32 OS_CloseAllFiles (void)

  Close all open files.
- int32 OS_CloseFileByName (const char *Filename)

  Close a file by filename.
- os_dirp_t OS_opendir (const char *path)
- int32 OS_closedir (os_dirp_t directory)
- void OS_rewinddir (os_dirp_t directory)
- os_dirent_t * OS_readdir (os_dirp_t directory)
- int32 OS_DirectoryOpen (uint32 *dir_id, const char *path)

  Opens a directory.
- int32 OS_DirectoryClose (uint32 dir_id)

  Closes an open directory.
- int32 OS_DirectoryRewind (uint32 dir_id)

Rewinds an open directory.

- int32 OS_DirectoryRead (uint32 dir_id, os_dirent_t *dirent)

  Reads the next name in the directory.
- int32 OS_mkdir (const char *path, uint32 access)

  Makes a new directory.
- int32 OS_rmdir (const char *path)

  Removes a directory from the file system.
- int32 OS_FileSysAddFixedMap (uint32 *filesys_id, const char *phys_path, const char *virt_path)

  Create a fixed mapping between an existing directory and a virtual OSAL mount point.
- int32 OS_mkfs (char *address, const char *devname, const char *volname, uint32 blocksize, uint32 numblocks)

  Makes a file system on the target.
- int32 OS_mount (const char *devname, const char *mountpoint)

  Mounts a file system.
- int32 OS_initfs (char *address, const char *devname, const char *volname, uint32 blocksize, uint32 numblocks)

  Initializes an existing file system.
- int32 OS_rmfs (const char *devname)

  Removes a file system.
- int32 OS_unmount (const char *mountpoint)

  Unmounts a mounted file system.
- int32 OS_fsBlocksFree (const char *name)

  Obtain number of blocks free.
- int32 OS_fsBytesFree (const char *name, uint64 *bytes_free)

  Obtains the number of free bytes in a volume.
- int32 OS_chkfs (const char *name, bool repair)

  Checks the health of a file system and repairs it if necessary.
- int32 OS_FS_GetPhysDriveName (char *PhysDriveName, const char *MountPoint)

  Obtains the physical drive name associated with a mount point.
- int32 OS_TranslatePath (const char *VirtualPath, char *LocalPath)

  Translates a OSAL Virtual file system path to a host Local path.
- int32 OS_GetFsInfo (os_fsinfo_t *filesys_info)

  Returns information about the file system.
- int32 OS_ShellOutputToFile (const char *Cmd, uint32 filedes)

  Executes the command and sends output to a file.

13.94.1 Macro Definition Documentation

13.94.1.1 ATA_DISK

```
#define ATA_DISK 3
```

Definition at line 36 of file osapi-os-filesys.h.

13.94.1.2 EEPROM_DISK

```
#define EEPROM_DISK 2
```

Definition at line 35 of file osapi-os-filesys.h.

13.94.1.3 FS_BASED

```
#define FS_BASED 0
```

Definition at line 33 of file osapi-os-filesys.h.

13.94.1.4 NUM_TABLE_ENTRIES

```
#define NUM_TABLE_ENTRIES 14
```

Definition at line 42 of file osapi-os-filesys.h.

13.94.1.5 OS_CHK_ONLY

```
#define OS_CHK_ONLY 0
```

Definition at line 30 of file osapi-os-filesys.h.

13.94.1.6 OS_DIRENTRY_NAME

```
#define OS_DIRENTRY_NAME(
              x ) ((x).FileName)
```

Definition at line 182 of file osapi-os-filesys.h.

13.94.1.7 OS_FILESTAT_EXEC

```
#define OS_FILESTAT_EXEC(
              x ) ((x).FileModeBits & OS_FILESTAT_MODE_EXEC)
```

Definition at line 158 of file osapi-os-filesys.h.

13.94.1.8 OS_FILESTAT_ISDIR

```
#define OS_FILESTAT_ISDIR(
              x ) ((x).FileModeBits & OS_FILESTAT_MODE_DIR)
```

Definition at line 157 of file osapi-os-filesys.h.

13.94.1.9 OS_FILESTAT_MODE

```
#define OS_FILESTAT_MODE(
              x ) ((x).FileModeBits)
```

Definition at line 156 of file osapi-os-filesys.h.

13.94.1.10 OS_FILESTAT_READ

```
#define OS_FILESTAT_READ(
              x ) ((x).FileModeBits & OS_FILESTAT_MODE_READ)
```

Definition at line 160 of file osapi-os-filesys.h.

13.94.1.11 OS_FILESTAT_SIZE

```
#define OS_FILESTAT_SIZE(
              x ) ((x).FileSize)
```

Definition at line 161 of file osapi-os-filesys.h.

13.94.1.12 OS_FILESTAT_TIME

```
#define OS_FILESTAT_TIME(
              x ) ((x).FileTime)
```

Definition at line 162 of file osapi-os-filesys.h.

13.94.1.13 OS_FILESTAT_WRITE

```
#define OS_FILESTAT_WRITE(
              x ) ((x).FileModeBits & OS_FILESTAT_MODE_WRITE)
```

Definition at line 159 of file osapi-os-filesys.h.

13.94.1.14 OS_FS_DEV_NAME_LEN

```
#define OS_FS_DEV_NAME_LEN 32
```

Definition at line 47 of file osapi-os-filesys.h.

13.94.1.15 OS_FS_ERR_DEVICE_NOT_FREE

```
#define OS_FS_ERR_DEVICE_NOT_FREE (-107)
```

Definition at line 63 of file osapi-os-filesys.h.

13.94.1.16 OS_FS_ERR_DRIVE_NOT_CREATED

```
#define OS_FS_ERR_DRIVE_NOT_CREATED (-106)
```

Definition at line 62 of file osapi-os-filesys.h.

13.94.1.17 OS_FS_ERR_INVALID_FD

```
#define OS_FS_ERR_INVALID_FD OS_ERR_INVALID_ID
```

Definition at line 75 of file osapi-os-filesys.h.

13.94.1.18 OS_FS_ERR_INVALID_POINTER

```
#define OS_FS_ERR_INVALID_POINTER OS_INVALID_POINTER
```

Definition at line 73 of file osapi-os-filesys.h.

13.94.1.19 OS_FS_ERR_NAME_TOO_LONG

```
#define OS_FS_ERR_NAME_TOO_LONG (-104)
```

Definition at line 61 of file osapi-os-filesys.h.

13.94.1.20 OS_FS_ERR_NO_FREE_FDS

```
#define OS_FS_ERR_NO_FREE_FDS OS_ERR_NO_FREE_IDS
```

Definition at line 74 of file osapi-os-filesys.h.

13.94.1.21 OS_FS_ERR_PATH_INVALID

```
#define OS_FS_ERR_PATH_INVALID (-108)
```

Definition at line 64 of file osapi-os-filesys.h.

13.94.1.22 OS_FS_ERR_PATH_TOO_LONG

```
#define OS_FS_ERR_PATH_TOO_LONG (-103)
```

Definition at line 60 of file osapi-os-filesys.h.

13.94.1.23 OS_FS_ERROR

```
#define OS_FS_ERROR OS_ERROR
```

Definition at line 72 of file osapi-os-filesys.h.

Referenced by CFE_ES_ShellOutputCommand(), and CFE_ES_StartApplications().

13.94.1.24 OS_FS_PHYS_NAME_LEN

```
#define OS_FS_PHYS_NAME_LEN 64
```

Definition at line 48 of file osapi-os-filesys.h.

13.94.1.25 OS_FS_SUCCESS

```
#define OS_FS_SUCCESS OS_SUCCESS
```

Definition at line 71 of file osapi-os-filesys.h.

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_InitializeFileSystems(), CFE_ES_ShellOutput↩Command(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CF↩E_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.94.1.26 OS_FS_UNIMPLEMENTED

```
#define OS_FS_UNIMPLEMENTED OS_ERR_NOT_IMPLEMENTED
```

Definition at line 76 of file osapi-os-filesys.h.

13.94.1.27 OS_FS_VOL_NAME_LEN

```
#define OS_FS_VOL_NAME_LEN 32
```

Definition at line 49 of file osapi-os-filesys.h.

13.94.1.28 OS_READ_ONLY

```
#define OS_READ_ONLY 0
```

Definition at line 22 of file osapi-os-filesys.h.

Referenced by CFE_ES_QueryAllCmd(), and CFE_ES_QueryAllTasksCmd().

13.94.1.29 OS_READ_WRITE

```
#define OS_READ_WRITE 2
```

Definition at line 24 of file osapi-os-filesys.h.

Referenced by CFE_ES_ShellOutputCommand().

13.94.1.30 OS_REPAIR

```
#define OS_REPAIR 1
```

Definition at line 31 of file osapi-os-filesys.h.

13.94.1.31 OS_SEEK_CUR

```
#define OS_SEEK_CUR 1
```

Definition at line 27 of file osapi-os-filesys.h.

13.94.1.32 OS_SEEK_END

```
#define OS_SEEK_END 2
```

Definition at line 28 of file osapi-os-filesys.h.

Referenced by CFE_ES_ShellOutputCommand().

13.94.1.33 OS_SEEK_SET

```
#define OS_SEEK_SET 0
```

Definition at line 26 of file osapi-os-filesys.h.

Referenced by CFE_ES_ListApplications(), CFE_ES_ListTasks(), and CFE_ES_ShellOutputCommand().

13.94.1.34 OS_WRITE_ONLY

```
#define OS_WRITE_ONLY 1
```

Definition at line 23 of file osapi-os-filesys.h.

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_QueryAllCmd(), CFE_ES↩_QueryAllTasksCmd(), CFE_ES_StopPerfDataCmd(), CFE_ES_SysLogDump(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.94.1.35 RAM_DISK

```
#define RAM_DISK 1
```

Definition at line 34 of file osapi-os-filesys.h.

13.94.2 Typedef Documentation

13.94.2.1 os_dirp_t

```
typedef void* os_dirp_t
```

Definition at line 176 of file osapi-os-filesys.h.

13.94.2.2 OS_FDTableEntry

```
typedef OS_file_prop_t OS_FDTableEntry
```

Definition at line 189 of file osapi-os-filesys.h.

13.94.2.3 os_fs_err_name_t

```
typedef os_err_name_t os_fs_err_name_t
```

Definition at line 91 of file osapi-os-filesys.h.

13.94.2.4 os_fshealth_t

```
typedef int32 os_fshealth_t
```

Definition at line 188 of file osapi-os-filesys.h.

13.94.3 Enumeration Type Documentation

13.94.3.1 anonymous enum

```
anonymous enum
```

Enumerator

OS_FILESTAT_MODE_EXEC	
OS_FILESTAT_MODE_WRITE	
OS_FILESTAT_MODE_READ	
OS_FILESTAT_MODE_DIR	

Definition at line 148 of file osapi-os-filesys.h.

13.94.4 Function Documentation

13.94.4.1 OS_chkfs()

```
int32 OS_chkfs (
            const char * name,
            bool repair )
```

Checks the drives for inconsistencies and optionally also repairs it

Note

not all operating systems implement this function

Parameters

in	*name*	The device/path to operate on
in	*repair*	Whether to also repair inconsistencies

Returns

on success, or appropriate error code OS_INVALID_POINTER if name is NULL OS_ERR_NOT_IMPLEMENTED if not supported OS_ERROR if the OS calls fail

13.94.4.2 OS_chmod()

```
int32 OS_chmod (
            const char * path,
            uint32 access )
```

Parameters

in	*path*	File to change
in	*access*	Desired access mode - OS_READ_ONLY, OS_WRITE_ONLY or OS_READ_WRITE

Note

Some file systems do not implement permissions

Returns

on success, or appropriate error code

13.94.4.3 OS_close()

```
int32 OS_close (
            uint32 filedes )
```

This closes regular file handles and any other file-like resource, such as network streams or pipes.

Parameters

in	*filedes*	The handle ID to operate on

Returns

on success, or appropriate error code OS_ERROR if file descriptor could not be closed OS_ERR_INVALID_ID if the file descriptor passed in is invalid

Referenced by CFE_ES_CleanupObjectCallback(), CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_PerfLogDump(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), CFE_ES_ShellOutputCommand(), CFE_ES_StartApplications(), CFE_ES_StopPerfDataCmd(), CFE_ES_SysLogDump(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.94.4.4 OS_CloseAllFiles()

```
int32 OS_CloseAllFiles (
            void  )
```

Closes All open files that were opened through the OSAL

Returns

on success, or appropriate error code OS_ERROR if one or more file close returned an error

13.94.4.5 OS_closedir()

```
int32 OS_closedir (
            os_dirp_t directory )
```

13.94.4.6 OS_CloseFileByName()

```
int32 OS_CloseFileByName (
            const char * Filename )
```

Allows a file to be closed by name. This will only work if the name passed in is the same name used to open the file.

Parameters

in	*Filename*	The file to close

Returns

on success, or appropriate error code OS_FS_ERR_PATH_INVALID if the file is not found OS_ERROR if the file close returned an error

13.94.4.7 OS_cp()

```
int32 OS_cp (
            const char * src,
            const char * dest )
```

Parameters

in	*src*	The source file to operate on
in	*dest*	The destination file

Returns

on success, or appropriate error code OS_ERROR if the file could not be accessed OS_INVALID_POINTER if src or dest are NULL OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_FS_ERR_PATH_TOO_LONG if the paths given are too long to be stored locally OS_FS_ERR_NAME_TOO_LONG if the dest name is too long to be stored locally

13.94.4.8 OS_creat()

```
int32 OS_creat (
            const char * path,
            int32 access )
```

Creates a file specified by const char *path, with read/write permissions by access. The file is also automatically opened by the create call.

Parameters

in	*path*	File name to create
in	*access*	Intended access mode - OS_WRITE_ONLY or OS_READ_WRITE

Note

Valid handle IDs are never negative. Failure of this call can be checked by testing if the result is less than 0.

Returns

file handle ID on success, or appropriate error code OS_INVALID_POINTER if path is NULL OS_FS_ERR_P↩ ATH_TOO_LONG if path exceeds the maximum number of chars OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_FS_ERR_NAME_TOO_LONG if the name of the file is too long OS_ERROR if permissions are unknown or OS call fails OS_ERR_NO_FREE_IDS if there are no free file descriptors left

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_QueryAllCmd(), CFE_ES↩ _QueryAllTasksCmd(), CFE_ES_ShellOutputCommand(), CFE_ES_StopPerfDataCmd(), CFE_ES_SysLogDump(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS_WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CFE_SB_Send↩ PipeInfo(), and CFE_SB_SendRtgInfo().

13.94.4.9 OS_DirectoryClose()

```
int32 OS_DirectoryClose (
            uint32 dir_id )
```

The directory referred to by dir_id will be closed

Parameters

in	*dir↩ _id*	The handle ID of the directory

Returns

on success, or appropriate error code

13.94.4.10 OS_DirectoryOpen()

```
int32 OS_DirectoryOpen (
```

```
            uint32 * dir_id,
            const char * path )
```

Prepares for reading the files within a directory

Parameters

out	*dir_id*	The handle ID of the directory
in	*path*	The directory to open

Returns

on success, or appropriate error code

13.94.4.11 OS_DirectoryRead()

```
int32 OS_DirectoryRead (
            uint32 dir_id,
            os_dirent_t * dirent )
```

Obtains directory entry data for the next file from an open directory

Parameters

in	*dir_id*	The handle ID of the directory
out	*dirent*	Buffer to store directory entry information

Returns

on success, or appropriate error code

13.94.4.12 OS_DirectoryRewind()

```
int32 OS_DirectoryRewind (
            uint32 dir_id )
```

Resets a directory read handle back to the first file.

Parameters

in	*dir_id*	The handle ID of the directory

Returns

on success, or appropriate error code

13.94.4.13 OS_FDGetInfo()

```
int32 OS_FDGetInfo (
            uint32 filedes,
            OS_file_prop_t * fd_prop )
```

Copies the information of the given file descriptor into a structure passed in

Parameters

in	*filedes*	The handle ID to operate on
out	*fd_prop*	Storage buffer for file information

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the file descriptor passed in is invalid OS_FS_SUCCESS if the copying was successfull

13.94.4.14 OS_FileOpenCheck()

```
int32 OS_FileOpenCheck (
            const char * Filename )
```

Parameters

in	*Filename*	The file to operate on

Returns

if the file is open, or appropriate error code OS_ERROR if the file is not open

13.94.4.15 OS_FileSysAddFixedMap()

```
int32 OS_FileSysAddFixedMap (
            uint32 * filesys_id,
            const char * phys_path,
            const char * virt_path )
```

This mimics the behavior of a "FS_BASED" entry in the VolumeTable but is registered at runtime. It is intended to be called by the PSP/BSP prior to starting the OSAL.

Parameters

out	*filesys_id*	An OSAL ID reflecting the file system
in	*phys_path*	The native system directory (an existing mount point)
in	*virt_path*	The virtual mount point of this filesystem

Returns

on success, or appropriate error code

13.94.4.16 OS_FS_GetPhysDriveName()

```
int32 OS_FS_GetPhysDriveName (
            char * PhysDriveName,
            const char * MountPoint )
```

Returns the name of the physical volume associated with the drive, when given the OSAL mount point of the drive

Parameters

out	*PhysDriveName*	Buffer to store physical drive name
in	*MountPoint*	OSAL mount point

Returns

on success, or appropriate error code OS_INVALID_POINTER if either parameter is NULL OS_ERROR if the mountpoint could not be found

13.94.4.17 OS_fsBlocksFree()

```
int32 OS_fsBlocksFree (
            const char * name )
```

Returns the number of free blocks in a volume

Parameters

in	*name*	The device/path to operate on

Returns

block count on success, or appropriate error code OS_INVALID_POINTER if name is NULL OS_FS_ERR_PATH_TOO_LONG if the name is too long OS_ERROR if the OS call failed

Referenced by CFE_ES_InitializeFileSystems().

13.94.4.18 OS_fsBytesFree()

```
int32 OS_fsBytesFree (
            const char * name,
            uint64 * bytes_free )
```

Returns the number of free bytes in a volume

Note

uses a 64 bit data type to support filesystems that are greater than 4 Gigabytes

Parameters

in	*name*	The device/path to operate on
out	*bytes_free*	The number of free bytes

Returns

on success, or appropriate error code OS_INVALID_POINTER if name is NULL OS_FS_ERR_PATH_TOO_L↩ ONG if the name is too long OS_ERROR if the OS call failed

13.94.4.19 OS_GetFsInfo()

```
int32 OS_GetFsInfo (
            os_fsinfo_t * filesys_info )
```

returns information about the file system in an os_fsinfo_t This includes the number of open files and file systems

Parameters

out	*filesys_info*	Buffer to store filesystem information

Returns

on success, or appropriate error code OS_INVALID_POINTER if filesys_info is NULL

13.94.4.20 OS_initfs()

```
int32 OS_initfs (
            char * address,
            const char * devname,
            const char * volname,
            uint32 blocksize,
            uint32 numblocks )
```

Initializes a file system on the target.

Returns

on success, or appropriate error code OS_INVALID_POINTER if devname or volname are NULL OS_FS_E↩ RR_PATH_TOO_LONG if the name is too long OS_FS_ERR_DEVICE_NOT_FREE if the volume table is full OS_FS_ERR_DRIVE_NOT_CREATED on error

Referenced by CFE_ES_InitializeFileSystems().

13.94.4.21 OS_lseek()

```
int32 OS_lseek (
            uint32 filedes,
            int32 offset,
            uint32 whence )
```

sets the read/write pointer to a specific offset in a specific file. Whence is either OS_SEEK_SET,OS_SEEK_CUR, or OS_SEEK_END

Parameters

in	*filedes*	The handle ID to operate on
in	*offset*	The file offset to seek to
in	*whence*	The reference point for offset

Returns

success, a non-negative byte offset from the beginning of the file OS_ERR_INVALID_ID if the file descriptor passed in is invalid OS_ERROR if OS call failed

Referenced by CFE_ES_ListApplications(), CFE_ES_ListTasks(), and CFE_ES_ShellOutputCommand().

13.94.4.22 OS_mkdir()

```
int32 OS_mkdir (
            const char * path,
            uint32 access )
```

makes a directory specified by path.

Parameters

in	*path*	The new directory name
in	*access*	The permissions for the directory (reserved for future use)

Note

current implementations do not utilize the "access" parameter. Applications should still pass the intended value (OS_READ_WRITE or OS_READ_ONLY) to be compatible with future implementations.

Returns

on success, or appropriate error code OS_INVALID_POINTER if path is NULL OS_FS_ERR_PATH_TOO_LONG if the path is too long to be stored locally OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_ERROR if the OS call fails

13.94.4.23 OS_mkfs()

```
int32 OS_mkfs (
            char * address,
            const char * devname,
            const char * volname,
            uint32 blocksize,
            uint32 numblocks )
```

Makes a file system on the target

Returns

on success, or appropriate error code OS_INVALID_POINTER if devname is NULL OS_FS_ERR_DRIVE_NO↩T_CREATED if the OS calls to create the the drive failed OS_FS_ERR_DEVICE_NOT_FREE if the volume table is full OS_FS_SUCCESS on creating the disk

Referenced by CFE_ES_InitializeFileSystems().

13.94.4.24 OS_mount()

```
int32 OS_mount (
            const char * devname,
            const char * mountpoint )
```

mounts a file system / block device at the given mount point

Returns

on success, or appropriate error code

Referenced by CFE_ES_InitializeFileSystems().

13.94.4.25 OS_mv()

```
int32 OS_mv (
            const char * src,
            const char * dest )
```

This first attempts to rename the file, which is faster if the source and destination reside on the same file system.

If this fails, it falls back to copying the file and removing the original.

Parameters

in	*src*	The source file to operate on
in	*dest*	The destination file

Returns

on success, or appropriate error code OS_ERROR if the file could not be renamed. OS_INVALID_POINTER if src or dest are NULL OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_FS_ERR_PATH_TOO_LONG if the paths given are too long to be stored locally OS_FS_ERR_NAME_TOO_LONG if the dest name is too long to be stored locally

13.94.4.26 OS_open()

```
int32 OS_open (
            const char * path,
            int32 access,
            uint32 mode )
```

Opens a file. access parameters are OS_READ_ONLY, OS_WRITE_ONLY, or OS_READ_WRITE

Parameters

in	*path*	File name to create
in	*access*	Intended access mode - OS_READ_ONLY, OS_WRITE_ONLY or OS_READ_WRITE
in	*mode*	The file permissions. This parameter is passed through to the native open call, but will be ignored. The file mode (or permissions) are ignored by the POSIX open call when the O_CREAT access flag is not passed in.

Note

Valid handle IDs are never negative. Failure of this call can be checked by testing if the result is less than 0.

Returns

file handle ID on success, or appropriate error code OS_INVALID_POINTER if path is NULL OS_FS_ERR_P↩ ATH_TOO_LONG if path exceeds the maximum number of chars OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_FS_ERR_NAME_TOO_LONG if the name of the file is too long OS_ERROR if permissions are unknown or OS call fails OS_ERR_NO_FREE_IDS if there are no free file descriptors left

Referenced by CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasksCmd(), and CFE_ES_StartApplications().

13.94.4.27 OS_opendir()

```
os_dirp_t OS_opendir (
            const char * path )
```

13.94.4.28 OS_read()

```
int32 OS_read (
            uint32 filedes,
            void * buffer,
            uint32 nbytes )
```

reads up to nbytes from a file, and puts them into buffer.

Parameters

in	*filedes*	The handle ID to operate on
out	*buffer*	Storage location for file data
in	*nbytes*	Maximum number of bytes to read

Note

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

Returns

non-negative byte count on success, or appropriate error code OS_INVALID_POINTER if buffer is a null pointer OS_ERROR if OS call failed OS_ERR_INVALID_ID if the file descriptor passed in is invalid

Referenced by CFE_ES_ShellOutputCommand(), and CFE_ES_StartApplications().

13.94.4.29 OS_readdir()

```
os_dirent_t* OS_readdir (
            os_dirp_t directory )
```

13.94.4.30 OS_remove()

```
int32 OS_remove (
            const char * path )
```

Removes a given filename from the drive

Note

Some file systems permit removal of open files while others do not. For portability, it is recommended that applications ensure the file is closed prior to removal.

Parameters

in	*path*	The file to operate on

Returns

on success, or appropriate error code OS_ERROR if there is no device or the driver returns error OS_INVALID↩ _POINTER if path is NULL OS_FS_ERR_PATH_TOO_LONG if path is too long to be stored locally OS_FS_E↩ RR_PATH_INVALID if path cannot be parsed OS_FS_ERR_NAME_TOO_LONG if the name of the file to remove is too long

Referenced by CFE_ES_AppCreate(), CFE_ES_LoadLibrary(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasks↩ Cmd(), and CFE_ES_ShellOutputCommand().

13.94.4.31 OS_rename()

```
int32 OS_rename (
            const char * old_filename,
            const char * new_filename )
```

Changes the name of a file, where the source and destination reside on the same file system.

Note

Some file systems permit renaming of open files while others do not. For portability, it is recommended that applications ensure the file is closed prior to rename.

Parameters

in	*old_filename*	The original filename
in	*new_filename*	The desired filename

Returns

on success, or appropriate error code OS_ERROR if the file could not be opened or renamed. OS_INVALID_↩ POINTER if old or new are NULL OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_FS_ERR_PAT↩ H_TOO_LONG if the paths given are too long to be stored locally OS_FS_ERR_NAME_TOO_LONG if the new name is too long to be stored locally

13.94.4.32 OS_rewinddir()

```
void OS_rewinddir (
            os_dirp_t directory )
```

13.94.4.33 OS_rmdir()

```
int32 OS_rmdir (
            const char * path )
```

Removes a directory from the structure. The directory must be empty prior to this operation.

Parameters

in	*path*	The directory to remove

Returns

on success, or appropriate error code OS_INVALID_POINTER if path is NULL OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_FS_ER_PATH_TOO_LONG OS_ERROR if the directory remove operation failed

13.94.4.34 OS_rmfs()

```
int32 OS_rmfs (
            const char * devname )
```

Returns

on success, or appropriate error code OS_INVALID_POINTER if devname is NULL OS_ERROR is the drive specified cannot be located OS_FS_SUCCESS on removing the disk

Referenced by CFE_ES_InitializeFileSystems().

13.94.4.35 OS_ShellOutputToFile()

```
int32 OS_ShellOutputToFile (
            const char * Cmd,
            uint32 filedes )
```

Takes a shell command in and writes the output of that command to the specified file The output file must be opened previously with write access (OS_WRITE_ONLY or OS_READ_WRITE).

Parameters

in	*Cmd*	Command to pass to shell
in	*filedes*	File to send output to.

Returns

on success, or appropriate error code OS_ERROR if the command was not executed properly OS_ERR_INVA↩ LID_ID if the file descriptor passed in is invalid

Referenced by CFE_ES_ShellOutputCommand().

13.94.4.36 OS_stat()

```
int32 OS_stat (
            const char * path,
            os_fstat_t * filestats )
```

returns information about a file or directory in a os_fs_stat structure

Parameters

in	*path*	The file to operate on
out	*filestats*	Buffer to store file information

Returns

on success, or appropriate error code OS_INVALID_POINTER if path or filestats is NULL OS_FS_ERR_PATH↩ _TOO_LONG if the path is too long to be stored locally OS_FS_ERR_NAME_TOO_LONG if the name of the file is too long to be stored OS_FS_ERR_PATH_INVALID if path cannot be parsed OS_ERROR if the OS call failed

Referenced by CFE_ES_ReloadApp().

13.94.4.37 OS_TimedRead()

```
int32 OS_TimedRead (
            uint32 filedes,
            void * buffer,
            uint32 nbytes,
            int32 timeout )
```

This implements a time-limited read and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports.

If data is immediately available on the file/socket, this will return that data along with the actual number of bytes that were immediately available. It will not block.

If no data is immediately available, this will wait up to the given timeout for data to appear. If no data appears within the timeout period, then this returns an error code (not zero).

In all cases this will return successfully as soon as at least 1 byte of actual data is available. It will not attempt to read the entire input buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

Parameters

in	*filedes*	The handle ID to operate on
in	*buffer*	Source location for file data
in	*nbytes*	Maximum number of bytes to read
in	*timeout*	Maximum time to wait, in milliseconds (OS_PEND = forever)

Returns

non-negative byte count on success, or appropriate error code Returns zero if the timeout period expired.

13.94.4.38 OS_TimedWrite()

```
int32 OS_TimedWrite (
            uint32 filedes,
            const void * buffer,
            uint32 nbytes,
            int32 timeout )
```

This implements a time-limited write and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports.

If output buffer space is immediately available on the file/socket, this will place data into the buffer and return the actual number of bytes that were queued for output. It will not block.

If no output buffer space is immediately available, this will wait up to the given timeout for space to become available. If no space becomes available within the timeout period, then this returns an error code (not zero).

In all cases this will return successfully as soon as at least 1 byte of actual data is output. It will *not* attempt to write the entire output buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

Parameters

in	*filedes*	The handle ID to operate on
in	*buffer*	Source location for file data
in	*nbytes*	Maximum number of bytes to read
in	*timeout*	Maximum time to wait, in milliseconds (OS_PEND = forever)

Returns

non-negative byte count on success, or appropriate error code Returns zero if the timeout period expired.

13.94.4.39 OS_TranslatePath()

```
int32 OS_TranslatePath (
            const char * VirtualPath,
            char * LocalPath )
```

Translates a virtual path to an actual system path name

Parameters

in	*VirtualPath*	OSAL virtual path name
out	*LocalPath*	Buffer to store native/translated path name

Returns

on success, or appropriate error code OS_INVALID_POINTER if either parameter is NULL

13.94.4.40 OS_unmount()

```
int32 OS_unmount (
            const char * mountpoint )
```

Note

Any open file descriptors referencing this file system should be closed prior to unmounting a drive

Returns

on success, or appropriate error code OS_INVALID_POINTER if name is NULL OS_FS_ERR_PATH_TOO_L↩ ONG if the absolute path given is too long OS_ERROR if the OS calls failed

Referenced by CFE_ES_InitializeFileSystems().

13.94.4.41 OS_write()

```
int32 OS_write (
            uint32 filedes,
            const void * buffer,
            uint32 nbytes )
```

writes to a file. copies up to a maximum of nbytes of buffer to the file described in filedes

Parameters

in	*filedes*	The handle ID to operate on
in	*buffer*	Source location for file data
in	*nbytes*	Maximum number of bytes to read

Note

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

Returns

non-negative byte count on success, or appropriate error code OS_INVALID_POINTER if buffer is NULL OS_E↩ RROR if OS call failed OS_ERR_INVALID_ID if the file descriptor passed in is invalid

Referenced by CFE_ES_DumpCDSRegistryCmd(), CFE_ES_ERLogDump(), CFE_ES_ListApplications(), CFE_ES_↩ ListResources(), CFE_ES_ListTasks(), CFE_ES_PerfLogDump(), CFE_ES_QueryAllCmd(), CFE_ES_QueryAllTasks↩ Cmd(), CFE_ES_ShellOutputCommand(), CFE_ES_SysLogDump(), CFE_EVS_WriteAppDataFileCmd(), CFE_EVS↩ _WriteLogDataFileCmd(), CFE_SB_SendMapInfo(), CFE_SB_SendPipeInfo(), and CFE_SB_SendRtgInfo().

13.95 osal/src/os/inc/osapi-os-loader.h File Reference

Data Structures

- struct OS_module_address_t
- struct OS_module_prop_t
- struct OS_static_symbol_record_t

Typedefs

- typedef OS_module_prop_t OS_module_record_t

Functions

- int32 OS_SymbolLookup (cpuaddr *symbol_address, const char *symbol_name)

  Find the Address of a Symbol.
- int32 OS_SymbolTableDump (const char *filename, uint32 size_limit)

  Dumps the system symbol table to a file.
- int32 OS_ModuleLoad (uint32 *module_id, const char *module_name, const char *filename)

  Loads an object file.
- int32 OS_ModuleUnload (uint32 module_id)

  Unloads the module file.
- int32 OS_ModuleInfo (uint32 module_id, OS_module_prop_t *module_info)

  Obtain information about a module.

13.95.1 Typedef Documentation

13.95.1.1 OS_module_record_t

```
typedef OS_module_prop_t OS_module_record_t
```

Definition at line 85 of file osapi-os-loader.h.

13.95.2 Function Documentation

13.95.2.1 OS_ModuleInfo()

```
int32 OS_ModuleInfo (
            uint32 module_id,
            OS_module_prop_t * module_info )
```

Returns information about the loadable module

Parameters

in	*module_id*	OSAL ID of the previously the loaded module
out	*module_info*	Buffer to store module information

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the module id invalid OS_INVALID_POINTER if the pointer to the ModuleInfo structure is invalid

Referenced by CFE_ES_GetAppInfoInternal().

13.95.2.2 OS_ModuleLoad()

```
int32 OS_ModuleLoad (
            uint32 * module_id,
            const char * module_name,
            const char * filename )
```

Loads an object file into the running operating system

Parameters

out	*module_id*	OSAL ID corresponding to the loaded module
in	*module_name*	Name of module
in	*filename*	File containing the object code to load

Returns

on success, or appropriate error code OS_ERROR if the module cannot be loaded OS_INVALID_POINTER if one of the parameters is NULL OS_ERR_NO_FREE_IDS if the module table is full OS_ERR_NAME_TAKEN if the name is in use

Referenced by CFE_ES_AppCreate(), and CFE_ES_LoadLibrary().

13.95.2.3 OS_ModuleUnload()

```
int32 OS_ModuleUnload (
            uint32 module_id )
```

Unloads the module file from the running operating system

Parameters

in	*module_id*	OSAL ID of the previously the loaded module

Returns

on success, or appropriate error code OS_ERROR if the module is invalid or cannot be unloaded

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanUpApp(), CFE_ES_CleanupObjectCallback(), and CFE_ES_LoadLibrary().

13.95.2.4 OS_SymbolLookup()

```
int32 OS_SymbolLookup (
            cpuaddr * symbol_address,
            const char * symbol_name )
```

This calls to the OS dynamic symbol lookup implementation, and/or checks a static symbol table for a matching symbol name.

The static table is intended to support embedded targets that do not have module loading capability or have it disabled.

Parameters

out	*symbol_address*	Set to the address of the symbol
in	*symbol_name*	Name of the symbol to look up

Returns

on success, or appropriate error code OS_ERROR if the symbol could not be found OS_INVALID_POINTER if one of the pointers passed in are NULL

Referenced by CFE_ES_AppCreate(), and CFE_ES_LoadLibrary().

13.95.2.5 OS_SymbolTableDump()

```
int32 OS_SymbolTableDump (
            const char * filename,
            uint32 size_limit )
```

Parameters

in	*filename*	File to write to
in	*size_limit*	Maximum number of bytes to write

Returns

on success, or appropriate error code OS_ERR_NOT_IMPLEMENTED if the system does not support this function OS_ERROR if the symbol table could not be read or dumped OS_INVALID_FILE if the file could not be opened or written

13.96 osal/src/os/inc/osapi-os-net.h File Reference

```
#include <osconfig.h>
```

Data Structures

- union OS_SockAddrData_t
- struct OS_SockAddr_t
- struct OS_socket_prop_t

Macros

- #define OS_SOCKADDR_MAX_LEN 28

Enumerations

- enum OS_SocketDomain_t { OS_SocketDomain_INVALID, OS_SocketDomain_INET, OS_SocketDomain_IN↩ ET6, OS_SocketDomain_MAX }
- enum OS_SocketType_t { OS_SocketType_INVALID, OS_SocketType_DATAGRAM, OS_SocketType_STREAM, OS_SocketType_MAX }

Functions

- int32 OS_SocketAddrInit (OS_SockAddr_t ∗Addr, OS_SocketDomain_t Domain)

  Initialize a socket address structure to hold an address of the given family.
- int32 OS_SocketAddrToString (char ∗buffer, uint32 buflen, const OS_SockAddr_t ∗Addr)

  Get a string representation of a network host address.
- int32 OS_SocketAddrFromString (OS_SockAddr_t ∗Addr, const char ∗string)

  Set a network host address from a string representation.
- int32 OS_SocketAddrGetPort (uint16 ∗PortNum, const OS_SockAddr_t ∗Addr)

  Get the port number of a network address.
- int32 OS_SocketAddrSetPort (OS_SockAddr_t ∗Addr, uint16 PortNum)

  Set the port number of a network address.
- int32 OS_SocketOpen (uint32 ∗sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type)

  Opens a socket.
- int32 OS_SocketBind (uint32 sock_id, const OS_SockAddr_t ∗Addr)

  Binds a socket to a given local address.
- int32 OS_SocketConnect (uint32 sock_id, const OS_SockAddr_t ∗Addr, int32 timeout)

  Connects a socket to a given remote address.
- int32 OS_SocketAccept (uint32 sock_id, uint32 ∗connsock_id, OS_SockAddr_t ∗Addr, int32 timeout)

  Waits for and accept the next incoming connection on the given socket.
- int32 OS_SocketRecvFrom (uint32 sock_id, void ∗buffer, uint32 buflen, OS_SockAddr_t ∗RemoteAddr, int32 timeout)

  Reads data from a message-oriented (datagram) socket.
- int32 OS_SocketSendTo (uint32 sock_id, const void ∗buffer, uint32 buflen, const OS_SockAddr_t ∗RemoteAddr)

  Sends data to a message-oriented (datagram) socket.
- int32 OS_SocketGetIdByName (uint32 ∗sock_id, const char ∗sock_name)

  Gets an OSAL ID from a given name.
- int32 OS_SocketGetInfo (uint32 sock_id, OS_socket_prop_t ∗sock_prop)

  Gets information about an OSAL Socket ID.
- int32 OS_NetworkGetID (void)

  Gets the network ID of the local machine.
- int32 OS_NetworkGetHostName (char ∗host_name, uint32 name_len)

  Gets the local machine network host name.

13.96.1 Macro Definition Documentation

13.96.1.1 OS_SOCKADDR_MAX_LEN

```
#define OS_SOCKADDR_MAX_LEN 28
```

Definition at line 37 of file osapi-os-net.h.

13.96.2 Enumeration Type Documentation

13.96.2.1 OS_SocketDomain_t

```
enum OS_SocketDomain_t
```

Enumerator

OS_SocketDomain_INVALID	
OS_SocketDomain_INET	IPv4 address family, most commonly used)
OS_SocketDomain_INET6	IPv6 address family, depends on OS/network stack support
OS_SocketDomain_MAX	

Definition at line 52 of file osapi-os-net.h.

13.96.2.2 OS_SocketType_t

```
enum OS_SocketType_t
```

Enumerator

OS_SocketType_INVALID	
OS_SocketType_DATAGRAM	A connectionless, message-oriented socket
OS_SocketType_STREAM	A stream-oriented socket with the concept of a connection
OS_SocketType_MAX	

Definition at line 60 of file osapi-os-net.h.

13.96.3 Function Documentation

13.96.3.1 OS_NetworkGetHostName()

```
int32 OS_NetworkGetHostName (
            char * host_name,
            uint32 name_len )
```

If configured in the underlying network stack, this function retrieves the local hostname of the system.

Parameters

out	*host_name*	Buffer to hold name information
in	*name_len*	Maximum length of host name buffer

Returns

if successful

13.96.3.2 OS_NetworkGetID()

```
int32 OS_NetworkGetID (
            void  )
```

The ID is an implementation-defined value and may not be consistent in meaning across different platform types.

Note

this API may be removed in a future version of OSAL due to inconsistencies between platforms.

Returns

ID or fixed value of -1 if the host id could not be found

Note it is not possible to differentiate between error codes and valid network IDs here. It is assumed, however, that -1 is never a valid ID.

13.96.3.3 OS_SocketAccept()

```
int32 OS_SocketAccept (
            uint32 sock_id,
            uint32 * connsock_id,
            OS_SockAddr_t * Addr,
            int32 timeout )
```

This is used for sockets operating in a "server" role. The socket must be a stream type (connection-oriented) and previously bound to a local address using OS_SocketBind(). This will block the caller up to the given timeout or until an incoming connection request occurs, whichever happens first.

The new stream connection is then returned to the caller and the original server socket ID can be reused for the next connection.

Parameters

in	*sock_id*	The server socket ID, previously bound using OS_SocketBind()
out	*connsock↩ _id*	The connection socket, a new ID that can be read/written
in	*Addr*	The remote address of the incoming connection
in	*timeout*	The maximum amount of time to wait, or OS_PEND to wait forever

Returns

if successful

13.96.3.4 OS_SocketAddrFromString()

```
int32 OS_SocketAddrFromString (
            OS_SockAddr_t * Addr,
            const char * string )
```

The specific format of the output string depends on the address family.

The address structure should have been previously initialized using OS_SocketAddrInit() to set the address family type.

Note

For IPv4, this would typically be the dotted-decimal format (X.X.X.X). It is up to the discretion of the underlying implementation whether to accept hostnames, as this depends on the availability of DNS services. Since many embedded deployments do not have name services, this should not be relied upon.

Parameters

out	*Addr*	The address buffer to initialize
in	*string*	The string to initialize the address from.

Returns

if successful

13.96.3.5 OS_SocketAddrGetPort()

```
int32 OS_SocketAddrGetPort (
            uint16 * PortNum,
            const OS_SockAddr_t * Addr )
```

For network prototcols that have the concept of a port number (such as TCP/IP and UDP/IP) this function gets the port number from the address structure.

Parameters

out	*PortNum*	Buffer to store the port number
in	*Addr*	The network address buffer

Returns

if successful

13.96.3.6 OS_SocketAddrInit()

```
int32 OS_SocketAddrInit (
            OS_SockAddr_t * Addr,
            OS_SocketDomain_t Domain )
```

The address is set to a suitable default value for the family.

Parameters

out	*Addr*	The address buffer to initialize
in	*Domain*	The address family

Returns

if successful

13.96.3.7 OS_SocketAddrSetPort()

```
int32 OS_SocketAddrSetPort (
            OS_SockAddr_t * Addr,
            uint16 PortNum )
```

For network prototcols that have the concept of a port number (such as TCP/IP and UDP/IP) this function sets the port number from the address structure.

Parameters

in	*PortNum*	The port number to set
out	*Addr*	The network address buffer

Returns

if successful

13.96.3.8 OS_SocketAddrToString()

```
int32 OS_SocketAddrToString (
            char * buffer,
            uint32 buflen,
            const OS_SockAddr_t * Addr )
```

The specific format of the output string depends on the address family.

This string should be suitable to pass back into OS_SocketAddrFromString() which should recreate the same network address, and it should also be meaningful to a user of printed or logged as a C string.

Note

For IPv4, this would typically be the dotted-decimal format (X.X.X.X).

Parameters

out	*buffer*	Buffer to hold the output string
in	*buflen*	Maximum length of the output string
in	*Addr*	The network address buffer to convert

Returns

if successful

13.96.3.9 OS_SocketBind()

```
int32 OS_SocketBind (
            uint32 sock_id,
            const OS_SockAddr_t * Addr )
```

The specified socket will be bound to the local address and port, if available.

If the socket is connectionless, then it only binds to the local address.

If the socket is connection-oriented (stream), then this will also put the socket into a listening state for incoming connections at the local address.

Parameters

in	*sock_id*	The socket ID
in	*Addr*	The local address to bind to

Returns

if successful

13.96.3.10 OS_SocketConnect()

```
int32 OS_SocketConnect (
            uint32 sock_id,
            const OS_SockAddr_t * Addr,
            int32 timeout )
```

The socket will be connected to the remote address and port, if available. This only applies to stream-oriented sockets. Calling this on a datagram socket will return an error (these sockets should use SendTo/RecvFrom).

Parameters

in	*sock_id*	The socket ID
in	*Addr*	The remote address to connect to
in	*timeout*	The maximum amount of time to wait, or OS_PEND to wait forever

Returns

if successful

13.96.3.11 OS_SocketGetIdByName()

```
int32 OS_SocketGetIdByName (
            uint32 * sock_id,
            const char * sock_name )
```

Note

OSAL Sockets use generated names according to the address and type.

See also

OS_SocketGetInfo()

Parameters

out	*sock_id*	Buffer to hold result
in	*sock_name*	Name of socket to find

Returns

if successful, or appropriate error code OS_INVALID_POINTER is id or name are NULL pointers OS_ERR_N↩ AME_TOO_LONG if the name given is to long to have been stored OS_ERR_NAME_NOT_FOUND if the name was not found in the table

13.96.3.12 OS_SocketGetInfo()

```
int32 OS_SocketGetInfo (
            uint32 sock_id,
            OS_socket_prop_t * sock_prop )
```

OSAL Sockets use generated names according to the address and type. This allows applications to find the name of a given socket.

Parameters

in	*sock_id*	The socket ID
out	*sock_prop*	Buffer to hold socket information

Returns

if successful, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid semaphore OS_↩ INVALID_POINTER if the count_prop pointer is null

13.96.3.13 OS_SocketOpen()

```
int32 OS_SocketOpen (
            uint32 * sock_id,
            OS_SocketDomain_t Domain,
            OS_SocketType_t Type )
```

A new, unconnected and unbound socket is allocated of the given domain and type.

Parameters

out	*sock↩ _id*	Buffer to hold the OSAL ID
in	*Domain*	The domain / address family of the socket (INET or INET6, etc)
in	*Type*	The type of the socket (STREAM or DATAGRAM)

Returns

if successful

13.96.3.14 OS_SocketRecvFrom()

```
int32 OS_SocketRecvFrom (
            uint32 sock_id,
            void * buffer,
            uint32 buflen,
            OS_SockAddr_t * RemoteAddr,
            int32 timeout )
```

If a message is already available on the socket, this should immediately return that data without blocking. Otherwise, it may block up to the given timeout.

Parameters

in	*sock_id*	The socket ID, previously bound using OS_SocketBind()
out	*buffer*	Pointer to message data receive buffer
in	*buflen*	The maximum length of the message data to receive
out	*RemoteAddr*	Buffer to store the remote network address (may be NULL)
in	*timeout*	The maximum amount of time to wait, or OS_PEND to wait forever

Returns

count of actual bytes received if successful, or an appropriate error code

13.96.3.15 OS_SocketSendTo()

```
int32 OS_SocketSendTo (
            uint32 sock_id,
            const void * buffer,
            uint32 buflen,
            const OS_SockAddr_t * RemoteAddr )
```

This sends data in a non-blocking mode. If the socket is not currently able to queue the message, such as if its outbound buffer is full, then this returns an error code.

Parameters

in	*sock_id*	The socket ID, which must be of the datagram type
in	*buffer*	Pointer to message data to send
in	*buflen*	The length of the message data to send
in	*RemoteAddr*	Buffer containing the remote network address to send to

Returns

count of actual bytes sent if successful, or an appropriate error code

13.97 osal/src/os/inc/osapi-os-timer.h File Reference

Data Structures

- struct OS_timer_prop_t
- struct OS_timebase_prop_t

Typedefs

- typedef void(* OS_TimerCallback_t) (uint32 timer_id)
- typedef uint32(* OS_TimerSync_t) (uint32 timer_id)

Functions

- int32 OS_TimeBaseCreate (uint32 *timebase_id, const char *timebase_name, OS_TimerSync_t external_sync)

  Create an abstract Time Base resource.
- int32 OS_TimeBaseSet (uint32 timebase_id, uint32 start_time, uint32 interval_time)

  Sets the tick period for simulated time base objects.
- int32 OS_TimeBaseDelete (uint32 timebase_id)

  Deletes a time base object.
- int32 OS_TimeBaseGetIdByName (uint32 *timebase_id, const char *timebase_name)

  Find the ID of an existing time base resource.
- int32 OS_TimeBaseGetInfo (uint32 timebase_id, OS_timebase_prop_t *timebase_prop)

  Obtain information about a timebase resource.
- int32 OS_TimeBaseGetFreeRun (uint32 timebase_id, uint32 *freerun_val)

  Read the value of the timebase free run counter.
- int32 OS_TimerCreate (uint32 *timer_id, const char *timer_name, uint32 *clock_accuracy, OS_TimerCallback_t callback_ptr)

  Create a timer object.
- int32 OS_TimerAdd (uint32 *timer_id, const char *timer_name, uint32 timebase_id, OS_ArgCallback_t callback_ptr, void *callback_arg)

  Add a timer object based on an existing TimeBase resource.
- int32 OS_TimerSet (uint32 timer_id, uint32 start_time, uint32 interval_time)

  Configures the expiration time of the timer object.
- int32 OS_TimerDelete (uint32 timer_id)

  Deletes a timer resource.
- int32 OS_TimerGetIdByName (uint32 *timer_id, const char *timer_name)

  Locate an existing timer resource by name.
- int32 OS_TimerGetInfo (uint32 timer_id, OS_timer_prop_t *timer_prop)

  Gets information about an existing timer.

13.97.1 Typedef Documentation

13.97.1.1 OS_TimerCallback_t

```
typedef void(* OS_TimerCallback_t) (uint32 timer_id)
```

Definition at line 25 of file osapi-os-timer.h.

13.97.1.2 OS_TimerSync_t

```
typedef uint32(* OS_TimerSync_t) (uint32 timer_id)
```

Definition at line 26 of file osapi-os-timer.h.

13.97.2 Function Documentation

13.97.2.1 OS_TimeBaseCreate()

```
int32 OS_TimeBaseCreate (
            uint32 * timebase_id,
            const char * timebase_name,
            OS_TimerSync_t external_sync )
```

An OSAL time base is an abstraction of a "timer tick" that can, in turn, be used for measurement of elapsed time between events.

Time bases can be simulated by the operating system using the OS kernel-provided timing facilities, or based on a hardware timing source if provided by the BSP.

A time base object has a servicing task associated with it, that runs at elevated priority and will thereby interrupt user-level tasks when timing ticks occur.

If the external_sync function is passed as NULL, the operating system kernel timing resources will be utilized for a simulated timer tick.

If the external_sync function is not NULL, this should point to a BSP-provided function that will block the calling task until the next tick occurs. This can be used for synchronizing with hardware events.

Note

When provisioning a tunable RTOS kernel, such as RTEMS, the kernel should be configured to support at least (OS_MAX_TASKS + OS_MAX_TIMEBASES) threads, to account for the helper threads associated with time base objects.

Parameters

out	*timebase_id*	An identifier corresponding to the timebase resource
in	*timebase_name*	The name of the time base
in	*external_sync*	A synchronization function for BSP hardware-based timer ticks

Returns

on success, or appropriate error code

Referenced by main().

13.97.2.2 OS_TimeBaseDelete()

```
int32 OS_TimeBaseDelete (
            uint32 timebase_id )
```

The helper task and any other resources associated with the time base abstraction will be freed.

Parameters

in	*timebase_id*	The timebase resource to delete

Returns

on success, or appropriate error code

13.97.2.3 OS_TimeBaseGetFreeRun()

```
int32 OS_TimeBaseGetFreeRun (
            uint32 timebase_id,
            uint32 * freerun_val )
```

Poll the timer free-running time counter in a lightweight fashion.

The free run count is a monotonically increasing value reflecting the total time elapsed since the timebase inception. Units are the same as the timebase itself, usually microseconds.

Applications may quickly and efficiently calculate relative time differences by polling this value and subtracting the previous counter value.

The absolute value of this counter is not relevant, because it will "roll over" after 2^32 units of time. For a timebase with microsecond units, this occurs approximately every 4294 seconds, or about 1.2 hours.

Note

To ensure consistency of results, the application should sample the value at a minimum of two times the roll over frequency, and calculate the difference between the consecutive samples.

Parameters

in	*timebase_id*	The timebase to operate on
out	*freerun_val*	Buffer to store the free run counter

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid timebase

13.97.2.4 OS_TimeBaseGetIdByName()

```
int32 OS_TimeBaseGetIdByName (
            uint32 * timebase_id,
            const char * timebase_name )
```

Given a time base name, find and output the ID associated with it.

Parameters

out	*timebase_id*	The timebase resource ID
in	*timebase_name*	The name of the timebase resource to find

Returns

on success, or appropriate error code OS_INVALID_POINTER if timebase_id or timebase_name are NULL pointers OS_ERR_NAME_TOO_LONG if the name given is to long to have been stored OS_ERR_NAME_NOT_FO↩ UND if the name was not found in the table

13.97.2.5 OS_TimeBaseGetInfo()

```
int32 OS_TimeBaseGetInfo (
            uint32 timebase_id,
            OS_timebase_prop_t * timebase_prop )
```

Fills the buffer referred to by the timebase_prop parameter with relevant information about the time base resource.

This function will pass back a pointer to structure that contains all of the relevant info(name and creator) about the specified timebase.

Parameters

in	*timebase_id*	The timebase resource ID
out	*timebase_prop*	Buffer to store timebase properties

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid timebase OS_IN↩ VALID_POINTER if the timebase_prop pointer is null

13.97.2.6 OS_TimeBaseSet()

```
int32 OS_TimeBaseSet (
            uint32 timebase_id,
            uint32 start_time,
            uint32 interval_time )
```

This sets the actual tick period for timing ticks that are simulated by the RTOS kernel (i.e. the "external_sync" parameter on the call to OS_TimeBaseCreate() is NULL).

The RTOS will be configured to wake up the helper thread at the requested interval.

This function has no effect for time bases that are using a BSP-provided external_sync function.

Parameters

in	*timebase_id*	The timebase resource to configure
in	*start_time*	The amount of delay for the first tick, in microseconds.
in	*interval_time*	The amount of delay between ticks, in microseconds.

Returns

on success, or appropriate error code

Referenced by main().

13.97.2.7 OS_TimerAdd()

```
int32 OS_TimerAdd (
            uint32 * timer_id,
            const char * timer_name,
            uint32 timebase_id,
            OS_ArgCallback_t callback_ptr,
            void * callback_arg )
```

A timer object is a resource that invokes the specified application-provided function upon timer expiration. Timers may be one-shot or periodic in nature.

This function uses an existing time base object to service this timer, which must exist prior to adding the timer. The precision of the timer is the same as that of the underlying time base object. Multiple timer objects can be created referring to a single time base object.

This routine also uses a different callback function prototype from OS_TimerCreate(), allowing a single opaque argument to be passed to the callback routine. The OSAL implementation does not use this parameter, and may be set NULL.

Parameters

out	*timer_id*	The resource ID of the timer object
in	*timer_name*	Name of the timer object
in	*timebase_id*	The time base resource to use as a reference
in	*callback_ptr*	Application-provided function to invoke
in	*callback_arg*	Opaque argument to pass to callback function

Returns

on success, or appropriate error code

13.97.2.8 OS_TimerCreate()

```
int32 OS_TimerCreate (
            uint32 * timer_id,
            const char * timer_name,
            uint32 * clock_accuracy,
            OS_TimerCallback_t callback_ptr )
```

A timer object is a resource that invokes the specified application-provided function upon timer expiration. Timers may be one-shot or periodic in nature.

This function creates a dedicated (hidden) time base object to service this timer, which is created and deleted with the timer object itself. The internal time base is configured for an OS simulated timer tick at the same interval as the timer.

Parameters

out	*timer_id*	The resource ID of the timer object
in	*timer_name*	Name of the timer object
out	*clock_accuracy*	Expected precision of the timer, in microseconds.
in	*callback_ptr*	Application-provided function to invoke

Returns

on success, or appropriate error code OS_INVALID_POINTER if any parameters are NULL OS_TIMER_ERR_↩ INVALID_ARGS if the callback function is not valid

13.97.2.9 OS_TimerDelete()

```
int32 OS_TimerDelete (
            uint32 timer_id )
```

The application callback associated with the timer will be stopped, and the resources freed for future use.

Parameters

in	*timer↩ _id*	The timer ID to operate on

Returns

on success, or appropriate error code

Referenced by CFE_ES_CleanupObjectCallback().

13.97.2.10 OS_TimerGetIdByName()

```
int32 OS_TimerGetIdByName (
            uint32 * timer_id,
            const char * timer_name )
```

Outputs the ID associated with the given timer, if it exists.

Parameters

out	*timer_id*	The timer ID corresponding to the name
in	*timer_name*	The timer name to find

Returns

on success, or appropriate error code OS_INVALID_POINTER if timer_id or timer_name are NULL pointers O↩ S_ERR_NAME_TOO_LONG if the name given is to long to have been stored OS_ERR_NAME_NOT_FOUND if

the name was not found in the table

13.97.2.11 OS_TimerGetInfo()

```
int32 OS_TimerGetInfo (
            uint32 timer_id,
            OS_timer_prop_t * timer_prop )
```

This function will populate structure with the relevant info (name and creator) about the specified timer.

Parameters

in	*timer_id*	The timer ID to operate on
out	*timer_prop*	Buffer containing timer properties

Returns

on success, or appropriate error code OS_ERR_INVALID_ID if the id passed in is not a valid timer OS_INVALI↩ D_POINTER if the timer_prop pointer is null

13.97.2.12 OS_TimerSet()

```
int32 OS_TimerSet (
            uint32 timer_id,
            uint32 start_time,
            uint32 interval_time )
```

Sets a timer to expire at the given start_time and interval_time. Units are the same as the underlying time base object. This is generally microseconds for RTOS-provided (simulated) time base objects, but may be different for BSP-provided time base objects.

For a "one-shot" timer, the start_time configures the expiration time, and the interval_time should be passed as zero to indicate the timer is not to be automatically reset.

For a periodic timer, the interval_time indicates the desired period between callbacks.

The start_time and interval_time should not both be zero.

Parameters

in	*timer_id*	The timer ID to operate on
in	*start_time*	Time to the first expiration
in	*interval_time*	Time between subsequent intervals

Returns

on success, or appropriate error code

13.98 osal/src/os/inc/osapi-version.h File Reference

Macros

- #define OS_MAJOR_VERSION 5
- #define OS_MINOR_VERSION 0
- #define OS_REVISION 7
- #define OS_MISSION_REV 0
- #define OSAL_API_VERSION ((OS_MAJOR_VERSION * 10000) + (OS_MINOR_VERSION * 100) + OS_REVISION)

13.98.1 Macro Definition Documentation

13.98.1.1 OS_MAJOR_VERSION

```
#define OS_MAJOR_VERSION 5
```

Definition at line 21 of file osapi-version.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.98.1.2 OS_MINOR_VERSION

```
#define OS_MINOR_VERSION 0
```

Definition at line 22 of file osapi-version.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.98.1.3 OS_MISSION_REV

```
#define OS_MISSION_REV 0
```

Definition at line 24 of file osapi-version.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.98.1.4 OS_REVISION

```
#define OS_REVISION 7
```

Definition at line 23 of file osapi-version.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.98.1.5 OSAL_API_VERSION

```
#define OSAL_API_VERSION ((OS_MAJOR_VERSION * 10000) + (OS_MINOR_VERSION * 100) + OS_REVISION)
```

Combine the revision components into a single value that application code can check against e.g. "#if OSAL_API_V↩ ERSION >= 40100" would check if some feature added in OSAL 4.1 is present.

Definition at line 30 of file osapi-version.h.

13.99 osal/src/os/inc/osapi.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "common_types.h"
#include "osapi-version.h"
#include "osconfig.h"
#include "osapi-os-core.h"
#include "osapi-os-filesys.h"
#include "osapi-os-net.h"
#include "osapi-os-loader.h"
#include "osapi-os-timer.h"
```

Macros

- #define OS_SUCCESS (0)
- #define OS_ERROR (-1)
- #define OS_INVALID_POINTER (-2)
- #define OS_ERROR_ADDRESS_MISALIGNED (-3)
- #define OS_ERROR_TIMEOUT (-4)
- #define OS_INVALID_INT_NUM (-5)
- #define OS_SEM_FAILURE (-6)
- #define OS_SEM_TIMEOUT (-7)
- #define OS_QUEUE_EMPTY (-8)
- #define OS_QUEUE_FULL (-9)
- #define OS_QUEUE_TIMEOUT (-10)
- #define OS_QUEUE_INVALID_SIZE (-11)
- #define OS_QUEUE_ID_ERROR (-12)
- #define OS_ERR_NAME_TOO_LONG (-13)

- #define OS_ERR_NO_FREE_IDS (-14)
- #define OS_ERR_NAME_TAKEN (-15)
- #define OS_ERR_INVALID_ID (-16)
- #define OS_ERR_NAME_NOT_FOUND (-17)
- #define OS_ERR_SEM_NOT_FULL (-18)
- #define OS_ERR_INVALID_PRIORITY (-19)
- #define OS_INVALID_SEM_VALUE (-20)
- #define OS_ERR_FILE (-27)
- #define OS_ERR_NOT_IMPLEMENTED (-28)
- #define OS_TIMER_ERR_INVALID_ARGS (-29)
- #define OS_TIMER_ERR_TIMER_ID (-30)
- #define OS_TIMER_ERR_UNAVAILABLE (-31)
- #define OS_TIMER_ERR_INTERNAL (-32)
- #define OS_ERR_OBJECT_IN_USE (-33)
- #define OS_ERR_BAD_ADDRESS (-34)
- #define OS_ERR_INCORRECT_OBJ_STATE (-35)
- #define OS_ERR_INCORRECT_OBJ_TYPE (-36)
- #define OS_ERR_STREAM_DISCONNECTED (-37)
- #define OS_PEND (-1)
- #define OS_CHECK (0)

13.99.1 Macro Definition Documentation

13.99.1.1 OS_CHECK

```
#define OS_CHECK (0)
```

Definition at line 82 of file osapi.h.

Referenced by CFE_SB_ReadQueue().

13.99.1.2 OS_ERR_BAD_ADDRESS

```
#define OS_ERR_BAD_ADDRESS (-34)
```

Definition at line 73 of file osapi.h.

13.99.1.3 OS_ERR_FILE

```
#define OS_ERR_FILE (-27)
```

Definition at line 66 of file osapi.h.

13.99.1.4 OS_ERR_INCORRECT_OBJ_STATE

```
#define OS_ERR_INCORRECT_OBJ_STATE (-35)
```

Definition at line 74 of file osapi.h.

13.99.1.5 OS_ERR_INCORRECT_OBJ_TYPE

```
#define OS_ERR_INCORRECT_OBJ_TYPE (-36)
```

Definition at line 75 of file osapi.h.

13.99.1.6 OS_ERR_INVALID_ID

```
#define OS_ERR_INVALID_ID (-16)
```

Definition at line 61 of file osapi.h.

13.99.1.7 OS_ERR_INVALID_PRIORITY

```
#define OS_ERR_INVALID_PRIORITY (-19)
```

Definition at line 64 of file osapi.h.

13.99.1.8 OS_ERR_NAME_NOT_FOUND

```
#define OS_ERR_NAME_NOT_FOUND (-17)
```

Definition at line 62 of file osapi.h.

13.99.1.9 OS_ERR_NAME_TAKEN

```
#define OS_ERR_NAME_TAKEN (-15)
```

Definition at line 60 of file osapi.h.

Referenced by CFE_SB_CreatePipe().

13.99.1.10 OS_ERR_NAME_TOO_LONG

```
#define OS_ERR_NAME_TOO_LONG (-13)
```

Definition at line 58 of file osapi.h.

13.99.1.11 OS_ERR_NO_FREE_IDS

```
#define OS_ERR_NO_FREE_IDS (-14)
```

Definition at line 59 of file osapi.h.

Referenced by CFE_SB_CreatePipe().

13.99.1.12 OS_ERR_NOT_IMPLEMENTED

```
#define OS_ERR_NOT_IMPLEMENTED (-28)
```

Definition at line 67 of file osapi.h.

13.99.1.13 OS_ERR_OBJECT_IN_USE

```
#define OS_ERR_OBJECT_IN_USE (-33)
```

Definition at line 72 of file osapi.h.

13.99.1.14 OS_ERR_SEM_NOT_FULL

```
#define OS_ERR_SEM_NOT_FULL (-18)
```

Definition at line 63 of file osapi.h.

13.99.1.15 OS_ERR_STREAM_DISCONNECTED

```
#define OS_ERR_STREAM_DISCONNECTED (-37)
```

Definition at line 76 of file osapi.h.

13.99.1.16 OS_ERROR

```
#define OS_ERROR (-1)
```

Definition at line 46 of file osapi.h.

Referenced by CFE_ES_CleanUpApp(), and CFE_ES_CleanupObjectCallback().

13.99.1.17 OS_ERROR_ADDRESS_MISALIGNED

```
#define OS_ERROR_ADDRESS_MISALIGNED (-3)
```

Definition at line 48 of file osapi.h.

13.99.1.18 OS_ERROR_TIMEOUT

```
#define OS_ERROR_TIMEOUT (-4)
```

Definition at line 49 of file osapi.h.

13.99.1.19 OS_INVALID_INT_NUM

```
#define OS_INVALID_INT_NUM (-5)
```

Definition at line 50 of file osapi.h.

13.99.1.20 OS_INVALID_POINTER

```
#define OS_INVALID_POINTER (-2)
```

Definition at line 47 of file osapi.h.

13.99.1.21 OS_INVALID_SEM_VALUE

```
#define OS_INVALID_SEM_VALUE (-20)
```

Definition at line 65 of file osapi.h.

13.99.1.22 OS_PEND

```
#define OS_PEND (-1)
```

Definition at line 81 of file osapi.h.

Referenced by CFE_SB_ReadQueue().

13.99.1.23 OS_QUEUE_EMPTY

```
#define OS_QUEUE_EMPTY (-8)
```

Definition at line 53 of file osapi.h.

Referenced by CFE_SB_ReadQueue().

13.99.1.24 OS_QUEUE_FULL

```
#define OS_QUEUE_FULL (-9)
```

Definition at line 54 of file osapi.h.

Referenced by CFE_SB_SendMsgFull().

13.99.1.25 OS_QUEUE_ID_ERROR

```
#define OS_QUEUE_ID_ERROR (-12)
```

Definition at line 57 of file osapi.h.

13.99.1.26 OS_QUEUE_INVALID_SIZE

```
#define OS_QUEUE_INVALID_SIZE (-11)
```

Definition at line 56 of file osapi.h.

13.99.1.27 OS_QUEUE_TIMEOUT

```
#define OS_QUEUE_TIMEOUT (-10)
```

Definition at line 55 of file osapi.h.

Referenced by CFE_SB_ReadQueue().

13.99.1.28 OS_SEM_FAILURE

```
#define OS_SEM_FAILURE (-6)
```

Definition at line 51 of file osapi.h.

13.99.1.29 OS_SEM_TIMEOUT

```
#define OS_SEM_TIMEOUT (-7)
```

Definition at line 52 of file osapi.h.

13.99.1.30 OS_SUCCESS

```
#define OS_SUCCESS (0)
```

Definition at line 45 of file osapi.h.

Referenced by CFE_ES_AppCreate(), CFE_ES_CleanupObjectCallback(), CFE_ES_CleanupTaskResources(), CF↩
E_ES_CreateChildTask(), CFE_ES_CreateObjects(), CFE_ES_DeleteChildTask(), CFE_ES_ExitChildTask(), CFE_↩
ES_GetAppIDInternal(), CFE_ES_GetAppInfoInternal(), CFE_ES_GetTaskInfo(), CFE_ES_HousekeepingCmd(), C↩
FE_ES_IncrementTaskCounter(), CFE_ES_InitializeCDS(), CFE_ES_InitializeFileSystems(), CFE_ES_LoadLibrary(), CFE_ES_LockCDSRegistry(), CFE_ES_LockSharedData(), CFE_ES_Main(), CFE_ES_ProcessCoreException(), C↩
FE_ES_RebuildCDSPool(), CFE_ES_RegisterApp(), CFE_ES_RegisterChildTask(), CFE_ES_ReloadApp(), CFE_↩
ES_ShellOutputCommand(), CFE_ES_UnlockCDSRegistry(), CFE_ES_UnlockSharedData(), CFE_ES_UpdateCD↩
SRegistry(), CFE_EVS_EarlyInit(), CFE_SB_CreatePipe(), CFE_SB_EarlyInit(), CFE_SB_GetPipeIdByName(), CF↩
E_SB_GetPipeName(), CFE_SB_LockSharedData(), CFE_SB_ReadQueue(), CFE_SB_SendMsgFull(), CFE_SB_↩
UnlockSharedData(), and main().

13.99.1.31 OS_TIMER_ERR_INTERNAL

```
#define OS_TIMER_ERR_INTERNAL (-32)
```

Definition at line 71 of file osapi.h.

13.99.1.32 OS_TIMER_ERR_INVALID_ARGS

```
#define OS_TIMER_ERR_INVALID_ARGS (-29)
```

Definition at line 68 of file osapi.h.

13.99.1.33 OS_TIMER_ERR_TIMER_ID

```
#define OS_TIMER_ERR_TIMER_ID (-30)
```

Definition at line 69 of file osapi.h.

13.99.1.34 OS_TIMER_ERR_UNAVAILABLE

```
#define OS_TIMER_ERR_UNAVAILABLE (-31)
```

Definition at line 70 of file osapi.h.

13.100 psp/fsw/inc/cfe_psp.h File Reference

```
#include "common_types.h"
#include "osapi.h"
```

Data Structures

- struct CFE_PSP_MemTable_t

Macros

- #define CFE_PSP_SUCCESS (0)
- #define CFE_PSP_ERROR (-1)
- #define CFE_PSP_INVALID_POINTER (-2)
- #define CFE_PSP_ERROR_ADDRESS_MISALIGNED (-3)
- #define CFE_PSP_ERROR_TIMEOUT (-4)
- #define CFE_PSP_INVALID_INT_NUM (-5)
- #define CFE_PSP_INVALID_MEM_ADDR (-21)
- #define CFE_PSP_INVALID_MEM_TYPE (-22)
- #define CFE_PSP_INVALID_MEM_RANGE (-23)
- #define CFE_PSP_INVALID_MEM_WORDSIZE (-24)
- #define CFE_PSP_INVALID_MEM_SIZE (-25)
- #define CFE_PSP_INVALID_MEM_ATTR (-26)
- #define CFE_PSP_ERROR_NOT_IMPLEMENTED (-27)

- #define CFE_PSP_INVALID_MODULE_NAME (-28)
- #define CFE_PSP_INVALID_MODULE_ID (-29)
- #define CFE_PSP_PANIC_STARTUP 1
- #define CFE_PSP_PANIC_VOLATILE_DISK 2
- #define CFE_PSP_PANIC_MEMORY_ALLOC 3
- #define CFE_PSP_PANIC_NONVOL_DISK 4
- #define CFE_PSP_PANIC_STARTUP_SEM 5
- #define CFE_PSP_PANIC_CORE_APP 6
- #define CFE_PSP_PANIC_GENERAL_FAILURE 7
- #define BUFF_SIZE 256
- #define SIZE_BYTE 1
- #define SIZE_HALF 2
- #define SIZE_WORD 3
- #define CFE_PSP_MEM_RAM 1
- #define CFE_PSP_MEM_EEPROM 2
- #define CFE_PSP_MEM_ANY 3
- #define CFE_PSP_MEM_INVALID 4
- #define CFE_PSP_MEM_ATTR_WRITE 0x01
- #define CFE_PSP_MEM_ATTR_READ 0x02
- #define CFE_PSP_MEM_ATTR_READWRITE 0x03
- #define CFE_PSP_MEM_SIZE_BYTE 0x01
- #define CFE_PSP_MEM_SIZE_WORD 0x02
- #define CFE_PSP_MEM_SIZE_DWORD 0x04
- #define CFE_PSP_MAJOR_VERSION (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.MajorVersion)
- #define CFE_PSP_MINOR_VERSION (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.MinorVersion)
- #define CFE_PSP_REVISION (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.Revision)
- #define CFE_PSP_MISSION_REV (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.MissionRev)

Reset Types

- #define CFE_PSP_RST_TYPE_PROCESSOR 1
- #define CFE_PSP_RST_TYPE_POWERON 2
- #define CFE_PSP_RST_TYPE_MAX 3

Reset Sub-Types

- #define CFE_PSP_RST_SUBTYPE_POWER_CYCLE 1
  Reset caused by power having been removed and restored.
- #define CFE_PSP_RST_SUBTYPE_PUSH_BUTTON 2
  Reset caused by reset button on the board having been pressed.
- #define CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND 3
  Reset was caused by a reset line having been stimulated by a hardware special command.
- #define CFE_PSP_RST_SUBTYPE_HW_WATCHDOG 4
  Reset was caused by a watchdog timer expiring.
- #define CFE_PSP_RST_SUBTYPE_RESET_COMMAND 5
  Reset was caused by cFE ES processing a Reset Command .
- #define CFE_PSP_RST_SUBTYPE_EXCEPTION 6
  Reset was caused by a Processor Exception.
- #define CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET 7
  Reset was caused in an unknown manner.
- #define CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET 8
  Reset was caused by a JTAG or BDM connection.
- #define CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET 9
  Reset reverted to a cFE POWERON due to a boot bank switch.
- #define CFE_PSP_RST_SUBTYPE_MAX 10
  Placeholder to indicate 1+ the maximum value that the PSP will ever use.

Functions

- void CFE_PSP_Main (void)
- void CFE_PSP_GetTime (OS_time_t *LocalTime)
- void CFE_PSP_Restart (uint32 resetType)
- uint32 CFE_PSP_GetRestartType (uint32 *restartSubType)
- void CFE_PSP_FlushCaches (uint32 type, cpuaddr address, uint32 size)
- uint32 CFE_PSP_GetProcessorId (void)
- uint32 CFE_PSP_GetSpacecraftId (void)
- uint32 CFE_PSP_Get_Timer_Tick (void)
- uint32 CFE_PSP_GetTimerTicksPerSecond (void)
- uint32 CFE_PSP_GetTimerLow32Rollover (void)
- void CFE_PSP_Get_Timebase (uint32 *Tbu, uint32 *Tbl)
- uint32 CFE_PSP_Get_Dec (void)
- int32 CFE_PSP_InitProcessorReservedMemory (uint32 RestartType)
- int32 CFE_PSP_GetCDSSize (uint32 *SizeOfCDS)
- int32 CFE_PSP_WriteToCDS (const void *PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes)
- int32 CFE_PSP_ReadFromCDS (void *PtrToDataToRead, uint32 CDSOffset, uint32 NumBytes)
- int32 CFE_PSP_GetResetArea (cpuaddr *PtrToResetArea, uint32 *SizeOfResetArea)
- int32 CFE_PSP_GetUserReservedArea (cpuaddr *PtrToUserArea, uint32 *SizeOfUserArea)
- int32 CFE_PSP_GetVolatileDiskMem (cpuaddr *PtrToVolDisk, uint32 *SizeOfVolDisk)
- int32 CFE_PSP_GetKernelTextSegmentInfo (cpuaddr *PtrToKernelSegment, uint32 *SizeOfKernelSegment)
- int32 CFE_PSP_GetCFETextSegmentInfo (cpuaddr *PtrToCFESegment, uint32 *SizeOfCFESegment)
- void CFE_PSP_WatchdogInit (void)
- void CFE_PSP_WatchdogEnable (void)
- void CFE_PSP_WatchdogDisable (void)
- void CFE_PSP_WatchdogService (void)
- uint32 CFE_PSP_WatchdogGet (void)
- void CFE_PSP_WatchdogSet (uint32 WatchdogValue)
- void CFE_PSP_Panic (int32 ErrorCode)
- int32 CFE_PSP_InitSSR (uint32 bus, uint32 device, char *DeviceName)
- int32 CFE_PSP_Decompress (char *srcFileName, char *dstFileName)
- void CFE_PSP_AttachExceptions (void)
- void CFE_PSP_SetDefaultExceptionEnvironment (void)
- int32 CFE_PSP_PortRead8 (cpuaddr PortAddress, uint8 *ByteValue)
- int32 CFE_PSP_PortWrite8 (cpuaddr PortAddress, uint8 ByteValue)
- int32 CFE_PSP_PortRead16 (cpuaddr PortAddress, uint16 *uint16Value)
- int32 CFE_PSP_PortWrite16 (cpuaddr PortAddress, uint16 uint16Value)
- int32 CFE_PSP_PortRead32 (cpuaddr PortAddress, uint32 *uint32Value)
- int32 CFE_PSP_PortWrite32 (cpuaddr PortAddress, uint32 uint32Value)
- int32 CFE_PSP_MemRead8 (cpuaddr MemoryAddress, uint8 *ByteValue)
- int32 CFE_PSP_MemWrite8 (cpuaddr MemoryAddress, uint8 ByteValue)
- int32 CFE_PSP_MemRead16 (cpuaddr MemoryAddress, uint16 *uint16Value)
- int32 CFE_PSP_MemWrite16 (cpuaddr MemoryAddress, uint16 uint16Value)
- int32 CFE_PSP_MemRead32 (cpuaddr MemoryAddress, uint32 *uint32Value)
- int32 CFE_PSP_MemWrite32 (cpuaddr MemoryAddress, uint32 uint32Value)
- int32 CFE_PSP_MemCpy (void *dest, const void *src, uint32 n)
- int32 CFE_PSP_MemSet (void *dest, uint8 value, uint32 n)
- int32 CFE_PSP_MemValidateRange (cpuaddr Address, uint32 Size, uint32 MemoryType)
- uint32 CFE_PSP_MemRanges (void)

- int32 CFE_PSP_MemRangeSet (uint32 RangeNum, uint32 MemoryType, cpuaddr StartAddr, uint32 Size, uint32 WordSize, uint32 Attributes)
- int32 CFE_PSP_MemRangeGet (uint32 RangeNum, uint32 *MemoryType, cpuaddr *StartAddr, uint32 *Size, uint32 *WordSize, uint32 *Attributes)
- int32 CFE_PSP_EepromWrite8 (cpuaddr MemoryAddress, uint8 ByteValue)
- int32 CFE_PSP_EepromWrite16 (cpuaddr MemoryAddress, uint16 uint16Value)
- int32 CFE_PSP_EepromWrite32 (cpuaddr MemoryAddress, uint32 uint32Value)
- int32 CFE_PSP_EepromWriteEnable (uint32 Bank)
- int32 CFE_PSP_EepromWriteDisable (uint32 Bank)
- int32 CFE_PSP_EepromPowerUp (uint32 Bank)
- int32 CFE_PSP_EepromPowerDown (uint32 Bank)

13.100.1 Macro Definition Documentation

13.100.1.1 BUFF_SIZE

```
#define BUFF_SIZE 256
```

Definition at line 85 of file cfe_psp.h.

13.100.1.2 CFE_PSP_ERROR

```
#define CFE_PSP_ERROR (-1)
```

Definition at line 54 of file cfe_psp.h.

Referenced by CFE_PSP_GetCDSSize(), CFE_PSP_GetCFETextSegmentInfo(), CFE_PSP_GetKernelTextSegmentInfo(), CFE_PSP_GetResetArea(), CFE_PSP_GetUserReservedArea(), CFE_PSP_GetVolatileDiskMem(), CFE_PSP_InitSSR(), CFE_PSP_ReadFromCDS(), and CFE_PSP_WriteToCDS().

13.100.1.3 CFE_PSP_ERROR_ADDRESS_MISALIGNED

```
#define CFE_PSP_ERROR_ADDRESS_MISALIGNED (-3)
```

Definition at line 56 of file cfe_psp.h.

13.100.1.4 CFE_PSP_ERROR_NOT_IMPLEMENTED

```
#define CFE_PSP_ERROR_NOT_IMPLEMENTED (-27)
```

Definition at line 65 of file cfe_psp.h.

Referenced by CFE_PSP_GetKernelTextSegmentInfo().

13.100.1.5 CFE_PSP_ERROR_TIMEOUT

```
#define CFE_PSP_ERROR_TIMEOUT (-4)
```

Definition at line 57 of file cfe_psp.h.

13.100.1.6 CFE_PSP_INVALID_INT_NUM

```
#define CFE_PSP_INVALID_INT_NUM (-5)
```

Definition at line 58 of file cfe_psp.h.

13.100.1.7 CFE_PSP_INVALID_MEM_ADDR

```
#define CFE_PSP_INVALID_MEM_ADDR (-21)
```

Definition at line 59 of file cfe_psp.h.

13.100.1.8 CFE_PSP_INVALID_MEM_ATTR

```
#define CFE_PSP_INVALID_MEM_ATTR (-26)
```

Definition at line 64 of file cfe_psp.h.

13.100.1.9 CFE_PSP_INVALID_MEM_RANGE

```
#define CFE_PSP_INVALID_MEM_RANGE (-23)
```

Definition at line 61 of file cfe_psp.h.

13.100.1.10 CFE_PSP_INVALID_MEM_SIZE

```
#define CFE_PSP_INVALID_MEM_SIZE (-25)
```

Definition at line 63 of file cfe_psp.h.

13.100.1.11 CFE_PSP_INVALID_MEM_TYPE

```
#define CFE_PSP_INVALID_MEM_TYPE (-22)
```

Definition at line 60 of file cfe_psp.h.

13.100.1.12 CFE_PSP_INVALID_MEM_WORDSIZE

```
#define CFE_PSP_INVALID_MEM_WORDSIZE (-24)
```

Definition at line 62 of file cfe_psp.h.

13.100.1.13 CFE_PSP_INVALID_MODULE_ID

```
#define CFE_PSP_INVALID_MODULE_ID (-29)
```

Definition at line 67 of file cfe_psp.h.

13.100.1.14 CFE_PSP_INVALID_MODULE_NAME

```
#define CFE_PSP_INVALID_MODULE_NAME (-28)
```

Definition at line 66 of file cfe_psp.h.

13.100.1.15 CFE_PSP_INVALID_POINTER

```
#define CFE_PSP_INVALID_POINTER (-2)
```

Definition at line 55 of file cfe_psp.h.

13.100.1.16 CFE_PSP_MAJOR_VERSION

```
#define CFE_PSP_MAJOR_VERSION (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.MajorVersion)
```

Definition at line 140 of file cfe_psp.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.100.1.17 CFE_PSP_MEM_ANY

```
#define CFE_PSP_MEM_ANY 3
```

Definition at line 95 of file cfe_psp.h.

Referenced by CFE_ES_ValidateHandle().

13.100.1.18 CFE_PSP_MEM_ATTR_READ

```
#define CFE_PSP_MEM_ATTR_READ 0x02
```

Definition at line 102 of file cfe_psp.h.

13.100.1.19 CFE_PSP_MEM_ATTR_READWRITE

```
#define CFE_PSP_MEM_ATTR_READWRITE 0x03
```

Definition at line 103 of file cfe_psp.h.

13.100.1.20 CFE_PSP_MEM_ATTR_WRITE

```
#define CFE_PSP_MEM_ATTR_WRITE 0x01
```

Definition at line 101 of file cfe_psp.h.

13.100.1.21 CFE_PSP_MEM_EEPROM

```
#define CFE_PSP_MEM_EEPROM 2
```

Definition at line 94 of file cfe_psp.h.

13.100.1.22 CFE_PSP_MEM_INVALID

```
#define CFE_PSP_MEM_INVALID 4
```

Definition at line 96 of file cfe_psp.h.

13.100.1.23 CFE_PSP_MEM_RAM

```
#define CFE_PSP_MEM_RAM 1
```

Definition at line 93 of file cfe_psp.h.

13.100.1.24 CFE_PSP_MEM_SIZE_BYTE

```
#define CFE_PSP_MEM_SIZE_BYTE 0x01
```

Definition at line 108 of file cfe_psp.h.

13.100.1.25 CFE_PSP_MEM_SIZE_DWORD

```
#define CFE_PSP_MEM_SIZE_DWORD 0x04
```

Definition at line 110 of file cfe_psp.h.

13.100.1.26 CFE_PSP_MEM_SIZE_WORD

```
#define CFE_PSP_MEM_SIZE_WORD 0x02
```

Definition at line 109 of file cfe_psp.h.

13.100.1.27 CFE_PSP_MINOR_VERSION

```
#define CFE_PSP_MINOR_VERSION (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.MinorVersion)
```

Definition at line 141 of file cfe_psp.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.100.1.28 CFE_PSP_MISSION_REV

```
#define CFE_PSP_MISSION_REV (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.MissionRev)
```

Definition at line 143 of file cfe_psp.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.100.1.29 CFE_PSP_PANIC_CORE_APP

```
#define CFE_PSP_PANIC_CORE_APP 6
```

Definition at line 79 of file cfe_psp.h.

Referenced by CFE_ES_CreateObjects().

13.100.1.30 CFE_PSP_PANIC_GENERAL_FAILURE

```
#define CFE_PSP_PANIC_GENERAL_FAILURE 7
```

Definition at line 80 of file cfe_psp.h.

13.100.1.31 CFE_PSP_PANIC_MEMORY_ALLOC

```
#define CFE_PSP_PANIC_MEMORY_ALLOC 3
```

Definition at line 76 of file cfe_psp.h.

Referenced by CFE_ES_SetupResetVariables().

13.100.1.32 CFE_PSP_PANIC_NONVOL_DISK

```
#define CFE_PSP_PANIC_NONVOL_DISK 4
```

Definition at line 77 of file cfe_psp.h.

13.100.1.33 CFE_PSP_PANIC_STARTUP

```
#define CFE_PSP_PANIC_STARTUP 1
```

Definition at line 74 of file cfe_psp.h.

13.100.1.34 CFE_PSP_PANIC_STARTUP_SEM

```
#define CFE_PSP_PANIC_STARTUP_SEM 5
```

Definition at line 78 of file cfe_psp.h.

Referenced by CFE_ES_Main().

13.100.1.35 CFE_PSP_PANIC_VOLATILE_DISK

```
#define CFE_PSP_PANIC_VOLATILE_DISK 2
```

Definition at line 75 of file cfe_psp.h.

Referenced by CFE_ES_InitializeFileSystems().

13.100.1.36 CFE_PSP_REVISION

```
#define CFE_PSP_REVISION (GLOBAL_PSP_CONFIGDATA.PSP_VersionInfo.Revision)
```

Definition at line 142 of file cfe_psp.h.

Referenced by CFE_ES_NoopCmd(), and CFE_ES_TaskInit().

13.100.1.37 CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET

```
#define CFE_PSP_RST_SUBTYPE_BANKSWITCH_RESET 9
```

Definition at line 135 of file cfe_psp.h.

13.100.1.38 CFE_PSP_RST_SUBTYPE_EXCEPTION

```
#define CFE_PSP_RST_SUBTYPE_EXCEPTION 6
```

Definition at line 132 of file cfe_psp.h.

Referenced by CFE_ES_ProcessCoreException().

13.100.1.39 CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND

```
#define CFE_PSP_RST_SUBTYPE_HW_SPECIAL_COMMAND 3
```

Definition at line 129 of file cfe_psp.h.

Referenced by CFE_ES_SetupResetVariables().

13.100.1.40 CFE_PSP_RST_SUBTYPE_HW_WATCHDOG

```
#define CFE_PSP_RST_SUBTYPE_HW_WATCHDOG 4
```

Definition at line 130 of file cfe_psp.h.

Referenced by CFE_ES_SetupResetVariables().

13.100.1.41 CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET

```
#define CFE_PSP_RST_SUBTYPE_HWDEBUG_RESET 8
```

Definition at line 134 of file cfe_psp.h.

13.100.1.42 CFE_PSP_RST_SUBTYPE_MAX

```
#define CFE_PSP_RST_SUBTYPE_MAX 10
```

Definition at line 136 of file cfe_psp.h.

13.100.1.43 CFE_PSP_RST_SUBTYPE_POWER_CYCLE

```
#define CFE_PSP_RST_SUBTYPE_POWER_CYCLE 1
```

Definition at line 127 of file cfe_psp.h.

Referenced by CFE_ES_SetupResetVariables().

13.100.1.44 CFE_PSP_RST_SUBTYPE_PUSH_BUTTON

```
#define CFE_PSP_RST_SUBTYPE_PUSH_BUTTON 2
```

Definition at line 128 of file cfe_psp.h.

13.100.1.45 CFE_PSP_RST_SUBTYPE_RESET_COMMAND

```
#define CFE_PSP_RST_SUBTYPE_RESET_COMMAND 5
```

Definition at line 131 of file cfe_psp.h.

Referenced by CFE_ES_ResetCFE().

13.100.1.46 CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET

```
#define CFE_PSP_RST_SUBTYPE_UNDEFINED_RESET 7
```

Definition at line 133 of file cfe_psp.h.

13.100.1.47 CFE_PSP_RST_TYPE_MAX

```
#define CFE_PSP_RST_TYPE_MAX 3
```

Placeholder to indicate 1+ the maximum value that the PSP will ever use.

Definition at line 119 of file cfe_psp.h.

13.100.1.48 CFE_PSP_RST_TYPE_POWERON

```
#define CFE_PSP_RST_TYPE_POWERON 2
```

All memory has been cleared

Definition at line 118 of file cfe_psp.h.

Referenced by CFE_ES_InitializeFileSystems(), CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), CFE_ES↩
_RestartCmd(), CFE_ES_SetupResetVariables(), CFE_EVS_EarlyInit(), CFE_PSP_InitCDS(), CFE_PSP_InitReset↩
Area(), CFE_PSP_InitUserReservedArea(), CFE_PSP_Restart(), and main().

13.100.1.49 CFE_PSP_RST_TYPE_PROCESSOR

```
#define CFE_PSP_RST_TYPE_PROCESSOR 1
```

Volatile disk, Critical Data Store and User Reserved memory could still be valid

Definition at line 117 of file cfe_psp.h.

Referenced by CFE_ES_ExitApp(), CFE_ES_InitializeFileSystems(), CFE_ES_ProcessCoreException(), CFE_ES_↩
ResetCFE(), CFE_ES_RestartCmd(), CFE_ES_SetupPerfVariables(), CFE_ES_SetupResetVariables(), CFE_ES_↩
StartApplications(), CFE_PSP_Restart(), and main().

13.100.1.50 CFE_PSP_SUCCESS

```
#define CFE_PSP_SUCCESS (0)
```

Definition at line 53 of file cfe_psp.h.

Referenced by CFE_ES_CDS_EarlyInit(), CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_ER↩ LogDump(), CFE_ES_GetCDSBlock(), CFE_ES_InitCDSRegistry(), CFE_ES_InitializeCDS(), CFE_ES_Initialize↩ FileSystems(), CFE_ES_PutCDSBlock(), CFE_ES_RebuildCDS(), CFE_ES_RebuildCDSPool(), CFE_ES_Setup↩ ResetVariables(), CFE_ES_TaskInit(), CFE_ES_ValidateCDS(), CFE_ES_ValidateHandle(), CFE_EVS_EarlyInit(), CFE_PSP_GetCDSSize(), CFE_PSP_GetCFETextSegmentInfo(), CFE_PSP_GetResetArea(), CFE_PSP_GetUser↩ ReservedArea(), CFE_PSP_GetVolatileDiskMem(), CFE_PSP_InitCDS(), CFE_PSP_InitProcessorReservedMemory(), CFE_PSP_InitResetArea(), CFE_PSP_InitUserReservedArea(), CFE_PSP_InitVolatileDiskMem(), CFE_PSP_Read↩ FromCDS(), CFE_PSP_WriteToCDS(), and main().

13.100.1.51 SIZE_BYTE

```
#define SIZE_BYTE 1
```

Definition at line 86 of file cfe_psp.h.

13.100.1.52 SIZE_HALF

```
#define SIZE_HALF 2
```

Definition at line 87 of file cfe_psp.h.

13.100.1.53 SIZE_WORD

```
#define SIZE_WORD 3
```

Definition at line 88 of file cfe_psp.h.

13.100.2 Function Documentation

13.100.2.1 CFE_PSP_AttachExceptions()

```
void CFE_PSP_AttachExceptions (
             void  )
```

Definition at line 94 of file cfe_psp_exception.c.

References OS_printf().

Referenced by CFE_ES_Main().

Here is the call graph for this function:

13.100.2.2 CFE_PSP_Decompress()

```
int32 CFE_PSP_Decompress (
             char * srcFileName,
             char * dstFileName )
```

13.100.2.3 CFE_PSP_EepromPowerDown()

```
int32 CFE_PSP_EepromPowerDown (
             uint32 Bank )
```

13.100.2.4 CFE_PSP_EepromPowerUp()

```
int32 CFE_PSP_EepromPowerUp (
             uint32 Bank )
```

13.100.2.5 CFE_PSP_EepromWrite16()

```
int32 CFE_PSP_EepromWrite16 (
            cpuaddr MemoryAddress,
            uint16 uint16Value )
```

13.100.2.6 CFE_PSP_EepromWrite32()

```
int32 CFE_PSP_EepromWrite32 (
            cpuaddr MemoryAddress,
            uint32 uint32Value )
```

13.100.2.7 CFE_PSP_EepromWrite8()

```
int32 CFE_PSP_EepromWrite8 (
            cpuaddr MemoryAddress,
            uint8 ByteValue )
```

13.100.2.8 CFE_PSP_EepromWriteDisable()

```
int32 CFE_PSP_EepromWriteDisable (
            uint32 Bank )
```

13.100.2.9 CFE_PSP_EepromWriteEnable()

```
int32 CFE_PSP_EepromWriteEnable (
            uint32 Bank )
```

13.100.2.10 CFE_PSP_FlushCaches()

```
void CFE_PSP_FlushCaches (
            uint32 type,
            cpuaddr address,
            uint32 size )
```

Definition at line 125 of file cfe_psp_support.c.

13.100.2.11 CFE_PSP_Get_Dec()

```
uint32 CFE_PSP_Get_Dec (
            void  )
```

Definition at line 185 of file cfe_psp_timer.c.

13.100.2.12 CFE_PSP_Get_Timebase()

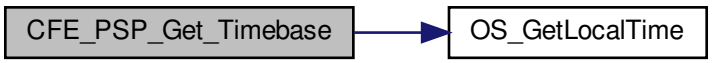
```
void CFE_PSP_Get_Timebase (
            uint32 * Tbu,
            uint32 * Tbl )
```

Definition at line 162 of file cfe_psp_timer.c.

References OS_time_t::microsecs, OS_GetLocalTime(), and OS_time_t::seconds.

Referenced by CFE_ES_PerfLogAdd().

Here is the call graph for this function:



13.100.2.13 CFE_PSP_Get_Timer_Tick()

```
uint32 CFE_PSP_Get_Timer_Tick (
            void  )
```

Definition at line 102 of file cfe_psp_timer.c.

13.100.2.14 CFE_PSP_GetCDSSize()

```
int32 CFE_PSP_GetCDSSize (
            uint32 * SizeOfCDS )
```

Definition at line 219 of file cfe_psp_memory.c.

References CFE_PSP_CDS_SIZE, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_CDS_EarlyInit().

13.100.2.15 CFE_PSP_GetCFETextSegmentInfo()

```
int32 CFE_PSP_GetCFETextSegmentInfo (
            cpuaddr * PtrToCFESegment,
            uint32 * SizeOfCFESegment )
```

Definition at line 781 of file cfe_psp_memory.c.

References _fini, _init, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_TaskInit().

13.100.2.16 CFE_PSP_GetKernelTextSegmentInfo()

```
int32 CFE_PSP_GetKernelTextSegmentInfo (
            cpuaddr * PtrToKernelSegment,
            uint32 * SizeOfKernelSegment )
```

Definition at line 753 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_ERROR_NOT_IMPLEMENTED, and NULL.

13.100.2.17 CFE_PSP_GetProcessorId()

```
uint32 CFE_PSP_GetProcessorId (
            void  )
```

Definition at line 147 of file cfe_psp_support.c.

References CFE_PSP_CpuId.

Referenced by CFE_SB_SendMsgFull(), and EVS_GenerateEventTelemetry().

13.100.2.18 CFE_PSP_GetResetArea()

```
int32 CFE_PSP_GetResetArea (
            cpuaddr * PtrToResetArea,
            uint32 * SizeOfResetArea )
```

Definition at line 434 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_RESET_AREA_SIZE, CFE_PSP_ResetAreaPtr, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_ERLogDump(), CFE_ES_SetupResetVariables(), and CFE_EVS_EarlyInit().

13.100.2.19 CFE_PSP_GetRestartType()

```
uint32 CFE_PSP_GetRestartType (
            uint32 * restartSubType )
```

13.100.2.20 CFE_PSP_GetSpacecraftId()

```
uint32 CFE_PSP_GetSpacecraftId (
            void  )
```

Definition at line 168 of file cfe_psp_support.c.

References CFE_PSP_SpacecraftId.

Referenced by EVS_GenerateEventTelemetry().

13.100.2.21 CFE_PSP_GetTime()

```
void CFE_PSP_GetTime (
            OS_time_t * LocalTime )
```

Definition at line 77 of file cfe_psp_timer.c.

References OS_GetLocalTime().

Here is the call graph for this function:

13.100.2.22 CFE_PSP_GetTimerLow32Rollover()

```
uint32 CFE_PSP_GetTimerLow32Rollover (
            void  )
```

Definition at line 144 of file cfe_psp_timer.c.

References CFE_PSP_TIMER_LOW32_ROLLOVER.

Referenced by CFE_ES_SetupPerfVariables().

13.100.2.23 CFE_PSP_GetTimerTicksPerSecond()

```
uint32 CFE_PSP_GetTimerTicksPerSecond (
            void  )
```

Definition at line 123 of file cfe_psp_timer.c.

References CFE_PSP_TIMER_TICKS_PER_SECOND.

Referenced by CFE_ES_SetupPerfVariables().

13.100.2.24 CFE_PSP_GetUserReservedArea()

```
int32 CFE_PSP_GetUserReservedArea (
            cpuaddr * PtrToUserArea,
            uint32 * SizeOfUserArea )
```

Definition at line 559 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_SUCCESS, CFE_PSP_USER_RESERVED_SIZE, CFE_PSP_User↩ ReservedAreaPtr, and NULL.

13.100.2.25 CFE_PSP_GetVolatileDiskMem()

```
int32 CFE_PSP_GetVolatileDiskMem (
            cpuaddr * PtrToVolDisk,
            uint32 * SizeOfVolDisk )
```

Definition at line 624 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_InitializeFileSystems().

13.100.2.26 CFE_PSP_InitProcessorReservedMemory()

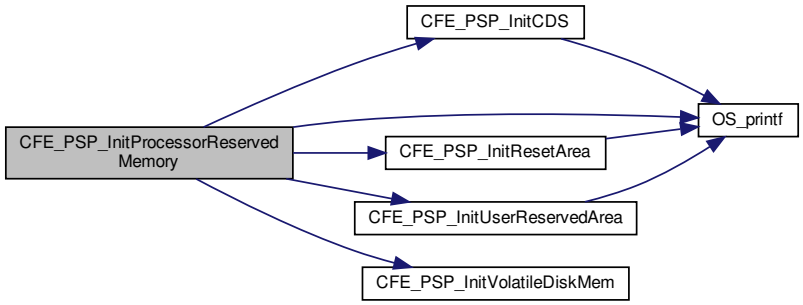
```
int32 CFE_PSP_InitProcessorReservedMemory (
            uint32 RestartType )
```

Definition at line 661 of file cfe_psp_memory.c.

References CFE_PSP_CDS_KEY_FILE, CFE_PSP_InitCDS(), CFE_PSP_InitResetArea(), CFE_PSP_InitUser↩ ReservedArea(), CFE_PSP_InitVolatileDiskMem(), CFE_PSP_RESERVED_KEY_FILE, CFE_PSP_RESET_KEY_F↩ ILE, CFE_PSP_SUCCESS, and OS_printf().

Referenced by main().

Here is the call graph for this function:



13.100.2.27 CFE_PSP_InitSSR()

```
int32 CFE_PSP_InitSSR (
            uint32 bus,
            uint32 device,
            char * DeviceName )
```

Definition at line 66 of file cfe_psp_ssr.c.

References CFE_PSP_ERROR.

13.100.2.28 CFE_PSP_Main()

```
void CFE_PSP_Main (
            void  )
```

13.100.2.29 CFE_PSP_MemCpy()

```
int32 CFE_PSP_MemCpy (
            void * dest,
            const void * src,
            uint32 n )
```

13.100.2.30 CFE_PSP_MemRangeGet()

```
int32 CFE_PSP_MemRangeGet (
            uint32 RangeNum,
            uint32 * MemoryType,
            cpuaddr * StartAddr,
            uint32 * Size,
            uint32 * WordSize,
            uint32 * Attributes )
```

13.100.2.31 CFE_PSP_MemRanges()

```
uint32 CFE_PSP_MemRanges (
            void  )
```

13.100.2.32 CFE_PSP_MemRangeSet()

```
int32 CFE_PSP_MemRangeSet (
            uint32 RangeNum,
            uint32 MemoryType,
            cpuaddr StartAddr,
            uint32 Size,
            uint32 WordSize,
            uint32 Attributes )
```

13.100.2.33 CFE_PSP_MemRead16()

```
int32 CFE_PSP_MemRead16 (
            cpuaddr MemoryAddress,
            uint16 * uint16Value )
```

13.100.2.34 CFE_PSP_MemRead32()

```
int32 CFE_PSP_MemRead32 (
            cpuaddr MemoryAddress,
            uint32 * uint32Value )
```

13.100.2.35 CFE_PSP_MemRead8()

```
int32 CFE_PSP_MemRead8 (
            cpuaddr MemoryAddress,
            uint8 * ByteValue )
```

13.100.2.36 CFE_PSP_MemSet()

```
int32 CFE_PSP_MemSet (
            void * dest,
            uint8 value,
            uint32 n )
```

13.100.2.37 CFE_PSP_MemValidateRange()

```
int32 CFE_PSP_MemValidateRange (
            cpuaddr Address,
            uint32 Size,
            uint32 MemoryType )
```

Referenced by CFE_ES_ValidateHandle().

13.100.2.38 CFE_PSP_MemWrite16()

```
int32 CFE_PSP_MemWrite16 (
            cpuaddr MemoryAddress,
            uint16 uint16Value )
```

13.100.2.39 CFE_PSP_MemWrite32()

```
int32 CFE_PSP_MemWrite32 (
            cpuaddr MemoryAddress,
            uint32 uint32Value )
```

13.100.2.40 CFE_PSP_MemWrite8()

```
int32 CFE_PSP_MemWrite8 (
            cpuaddr MemoryAddress,
            uint8 ByteValue )
```

13.100.2.41 CFE_PSP_Panic()

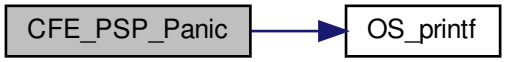
```
void CFE_PSP_Panic (
            int32 ErrorCode )
```

Definition at line 104 of file cfe_psp_support.c.

References OS_printf().

Referenced by CFE_ES_CreateObjects(), CFE_ES_InitializeFileSystems(), CFE_ES_Main(), CFE_ES_SetupReset↩Variables(), and main().

Here is the call graph for this function:



13.100.2.42 CFE_PSP_PortRead16()

```
int32 CFE_PSP_PortRead16 (
            cpuaddr PortAddress,
            uint16 * uint16Value )
```

13.100.2.43 CFE_PSP_PortRead32()

```
int32 CFE_PSP_PortRead32 (
            cpuaddr PortAddress,
            uint32 * uint32Value )
```

13.100.2.44 CFE_PSP_PortRead8()

```
int32 CFE_PSP_PortRead8 (
            cpuaddr PortAddress,
            uint8 * ByteValue )
```

13.100.2.45 CFE_PSP_PortWrite16()

```
int32 CFE_PSP_PortWrite16 (
            cpuaddr PortAddress,
            uint16 uint16Value )
```

13.100.2.46 CFE_PSP_PortWrite32()

```
int32 CFE_PSP_PortWrite32 (
            cpuaddr PortAddress,
            uint32 uint32Value )
```

13.100.2.47 CFE_PSP_PortWrite8()

```
int32 CFE_PSP_PortWrite8 (
            cpuaddr PortAddress,
            uint8 ByteValue )
```

13.100.2.48 CFE_PSP_ReadFromCDS()

```
int32 CFE_PSP_ReadFromCDS (
            void * PtrToDataToRead,
            uint32 CDSOffset,
            uint32 NumBytes )
```

Definition at line 292 of file cfe_psp_memory.c.

References CFE_PSP_CDS_SIZE, CFE_PSP_CDSPtr, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCD↩SBlock(), CFE_ES_RebuildCDS(), CFE_ES_RebuildCDSPool(), and CFE_ES_ValidateCDS().

13.100.2.49 CFE_PSP_Restart()

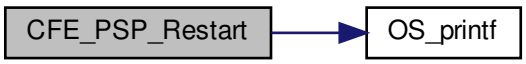
```
void CFE_PSP_Restart (
             uint32 resetType )
```

Definition at line 70 of file cfe_psp_support.c.

References CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, and OS_printf().

Referenced by CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), and CFE_ES_SetupResetVariables().

Here is the call graph for this function:



13.100.2.50 CFE_PSP_SetDefaultExceptionEnvironment()

```
void CFE_PSP_SetDefaultExceptionEnvironment (
             void  )
```

Definition at line 143 of file cfe_psp_exception.c.

Referenced by CFE_ES_RegisterApp(), and CFE_ES_RegisterChildTask().

13.100.2.51 CFE_PSP_WatchdogDisable()

```
void CFE_PSP_WatchdogDisable (
             void  )
```

Definition at line 114 of file cfe_psp_watchdog.c.

13.100.2.52 CFE_PSP_WatchdogEnable()

```
void CFE_PSP_WatchdogEnable (
             void  )
```

Definition at line 98 of file cfe_psp_watchdog.c.

13.100.2.53 CFE_PSP_WatchdogGet()

```
uint32 CFE_PSP_WatchdogGet (
            void  )
```

Definition at line 156 of file cfe_psp_watchdog.c.

References CFE_PSP_WatchdogValue.

13.100.2.54 CFE_PSP_WatchdogInit()

```
void CFE_PSP_WatchdogInit (
            void  )
```

Definition at line 75 of file cfe_psp_watchdog.c.

References CFE_PSP_WatchdogValue.

13.100.2.55 CFE_PSP_WatchdogService()

```
void CFE_PSP_WatchdogService (
            void  )
```

Definition at line 135 of file cfe_psp_watchdog.c.

13.100.2.56 CFE_PSP_WatchdogSet()

```
void CFE_PSP_WatchdogSet (
            uint32 WatchdogValue )
```

Definition at line 177 of file cfe_psp_watchdog.c.

References CFE_PSP_WatchdogValue.

13.100.2.57 CFE_PSP_WriteToCDS()

```
int32 CFE_PSP_WriteToCDS (
            const void * PtrToDataToWrite,
            uint32 CDSOffset,
            uint32 NumBytes )
```

Definition at line 249 of file cfe_psp_memory.c.

References CFE_PSP_CDS_SIZE, CFE_PSP_CDSPtr, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_InitCDSRegistry(), CFE_ES_↩ InitializeCDS(), CFE_ES_PutCDSBlock(), CFE_ES_RebuildCDSPool(), and CFE_ES_UpdateCDSRegistry().

13.101 psp/fsw/inc/cfe_psp_configdata.h File Reference

```
#include <osapi.h>
#include <cfe_psp.h>
```

Data Structures

- struct CFE_PSP_VersionInfo_t
- struct Target_PspConfigData

Variables

- Target_PspConfigData GLOBAL_PSP_CONFIGDATA
- CFE_PSP_MemTable_t CFE_PSP_MemoryTable [ ]
- OS_VolumeInfo_t OS_VolumeTable [ ]

13.101.1 Detailed Description

Created on: Dec 31, 2014 Author: joseph.p.hickey@nasa.gov

13.101.2 Variable Documentation

13.101.2.1 CFE_PSP_MemoryTable

```
CFE_PSP_MemTable_t CFE_PSP_MemoryTable[]
```

Extern reference to the psp memory table Allows the actual instantiation to be done outside this module

Definition at line 46 of file cfe_psp_memtab.c.

13.101.2.2 GLOBAL_PSP_CONFIGDATA

```
Target_PspConfigData GLOBAL_PSP_CONFIGDATA
```

Extern reference to psp config struct. Allows the actual instantiation to be done outside this module

13.101.2.3 OS_VolumeTable

```
OS_VolumeInfo_t OS_VolumeTable[]
```

Extern reference to the psp volume table Allows the actual instantiation to be done outside this module

Definition at line 63 of file cfe_psp_voltab.c.

13.102 psp/fsw/pc-linux/src/cfe_psp_exception.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
#include "cfe_psp_config.h"
#include <target_config.h>
```

Macros

- #define CFE_PSP_ES_EXCEPTION_FUNCTION (*GLOBAL_CONFIGDATA.CfeConfig->SystemExceptionI↩ SR)

Functions

- void CFE_PSP_ExceptionHook (int task_id, int vector, uint8 *pEsf)
- void CFE_PSP_AttachExceptions (void)
- void CFE_PSP_SetDefaultExceptionEnvironment (void)

Variables

- CFE_PSP_ExceptionContext_t CFE_PSP_ExceptionContext
- char CFE_PSP_ExceptionReasonString [256]

13.102.1 Macro Definition Documentation

13.102.1.1 CFE_PSP_ES_EXCEPTION_FUNCTION

```
#define CFE_PSP_ES_EXCEPTION_FUNCTION (*GLOBAL_CONFIGDATA.CfeConfig->SystemExceptionISR)
```

Definition at line 51 of file cfe_psp_exception.c.

Referenced by CFE_PSP_ExceptionHook().

13.102.2 Function Documentation

13.102.2.1 CFE_PSP_AttachExceptions()

```
void CFE_PSP_AttachExceptions (
            void  )
```

Definition at line 94 of file cfe_psp_exception.c.

References OS_printf().

Referenced by CFE_ES_Main().

Here is the call graph for this function:

13.102.2.2 CFE_PSP_ExceptionHook()

```
void CFE_PSP_ExceptionHook (
            int task_id,
            int vector,
            uint8 * pEsf )
```

Definition at line 108 of file cfe_psp_exception.c.

References CFE_PSP_ES_EXCEPTION_FUNCTION, CFE_PSP_ExceptionContext, and CFE_PSP_Exception↩ ReasonString.

13.102.2.3 CFE_PSP_SetDefaultExceptionEnvironment()

```
void CFE_PSP_SetDefaultExceptionEnvironment (
            void  )
```

Definition at line 143 of file cfe_psp_exception.c.

Referenced by CFE_ES_RegisterApp(), and CFE_ES_RegisterChildTask().

13.102.3 Variable Documentation

13.102.3.1 CFE_PSP_ExceptionContext

```
CFE_PSP_ExceptionContext_t CFE_PSP_ExceptionContext
```

Definition at line 69 of file cfe_psp_exception.c.

Referenced by CFE_PSP_ExceptionHook().

13.102.3.2 CFE_PSP_ExceptionReasonString

```
char CFE_PSP_ExceptionReasonString[256]
```

Definition at line 70 of file cfe_psp_exception.c.

Referenced by CFE_PSP_ExceptionHook().

13.103 psp/fsw/pc-linux/src/cfe_psp_memory.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <fcntl.h>
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
#include "cfe_psp_config.h"
#include <target_config.h>
```

Macros

- #define CFE_PSP_CDS_KEY_FILE ".cdskeyfile"
- #define CFE_PSP_RESET_KEY_FILE ".resetkeyfile"
- #define CFE_PSP_RESERVED_KEY_FILE ".reservedkeyfile"
- #define CFE_PSP_CDS_SIZE (GLOBAL_CONFIGDATA.CfeConfig->CdsSize)
- #define CFE_PSP_RESET_AREA_SIZE (GLOBAL_CONFIGDATA.CfeConfig->ResetAreaSize)
- #define CFE_PSP_USER_RESERVED_SIZE (GLOBAL_CONFIGDATA.CfeConfig->UserReservedSize)

Functions

- int32 CFE_PSP_InitCDS (uint32 RestartType)
- int32 CFE_PSP_InitResetArea (uint32 RestartType)
- int32 CFE_PSP_InitVolatileDiskMem (uint32 RestartType)
- int32 CFE_PSP_InitUserReservedArea (uint32 RestartType)
- void CFE_PSP_DeleteCDS (void)
- int32 CFE_PSP_GetCDSSize (uint32 *SizeOfCDS)
- int32 CFE_PSP_WriteToCDS (const void *PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes)
- int32 CFE_PSP_ReadFromCDS (void *PtrToDataToRead, uint32 CDSOffset, uint32 NumBytes)
- void CFE_PSP_DeleteResetArea (void)
- int32 CFE_PSP_GetResetArea (cpuaddr *PtrToResetArea, uint32 *SizeOfResetArea)
- void CFE_PSP_DeleteUserReservedArea (void)
- int32 CFE_PSP_GetUserReservedArea (cpuaddr *PtrToUserArea, uint32 *SizeOfUserArea)
- int32 CFE_PSP_GetVolatileDiskMem (cpuaddr *PtrToVolDisk, uint32 *SizeOfVolDisk)
- int32 CFE_PSP_InitProcessorReservedMemory (uint32 RestartType)
- void CFE_PSP_DeleteProcessorReservedMemory (void)
- int32 CFE_PSP_GetKernelTextSegmentInfo (cpuaddr *PtrToKernelSegment, uint32 *SizeOfKernelSegment)
- int32 CFE_PSP_GetCFETextSegmentInfo (cpuaddr *PtrToCFESegment, uint32 *SizeOfCFESegment)

Variables

- unsigned int _init
- unsigned int _fini
- uint8 * CFE_PSP_CDSPtr = 0
- uint8 * CFE_PSP_ResetAreaPtr = 0
- uint8 * CFE_PSP_UserReservedAreaPtr = 0
- int ResetAreaShmId
- int CDSShmId
- int UserShmId

13.103.1 Macro Definition Documentation

13.103.1.1 CFE_PSP_CDS_KEY_FILE

```
#define CFE_PSP_CDS_KEY_FILE ".cdskeyfile"
```

Definition at line 66 of file cfe_psp_memory.c.

Referenced by CFE_PSP_InitCDS(), and CFE_PSP_InitProcessorReservedMemory().

13.103.1.2 CFE_PSP_CDS_SIZE

```
#define CFE_PSP_CDS_SIZE (GLOBAL_CONFIGDATA.CfeConfig->CdsSize)
```

Definition at line 76 of file cfe_psp_memory.c.

Referenced by CFE_PSP_GetCDSSize(), CFE_PSP_InitCDS(), CFE_PSP_ReadFromCDS(), and CFE_PSP_Write↩ ToCDS().

13.103.1.3 CFE_PSP_RESERVED_KEY_FILE

```
#define CFE_PSP_RESERVED_KEY_FILE ".reservedkeyfile"
```

Definition at line 68 of file cfe_psp_memory.c.

Referenced by CFE_PSP_InitProcessorReservedMemory(), and CFE_PSP_InitUserReservedArea().

13.103.1.4 CFE_PSP_RESET_AREA_SIZE

```
#define CFE_PSP_RESET_AREA_SIZE (GLOBAL_CONFIGDATA.CfeConfig->ResetAreaSize)
```

Definition at line 77 of file cfe_psp_memory.c.

Referenced by CFE_PSP_GetResetArea(), and CFE_PSP_InitResetArea().

13.103.1.5 CFE_PSP_RESET_KEY_FILE

```
#define CFE_PSP_RESET_KEY_FILE ".resetkeyfile"
```

Definition at line 67 of file cfe_psp_memory.c.

Referenced by CFE_PSP_InitProcessorReservedMemory(), and CFE_PSP_InitResetArea().

13.103.1.6 CFE_PSP_USER_RESERVED_SIZE

```
#define CFE_PSP_USER_RESERVED_SIZE (GLOBAL_CONFIGDATA.CfeConfig->UserReservedSize)
```

Definition at line 78 of file cfe_psp_memory.c.

Referenced by CFE_PSP_GetUserReservedArea(), and CFE_PSP_InitUserReservedArea().

13.103.2 Function Documentation

13.103.2.1 CFE_PSP_DeleteCDS()

```
void CFE_PSP_DeleteCDS (
             void  )
```

Definition at line 185 of file cfe_psp_memory.c.

References CDSShmId.

Referenced by CFE_PSP_DeleteProcessorReservedMemory().

13.103.2.2 CFE_PSP_DeleteProcessorReservedMemory()

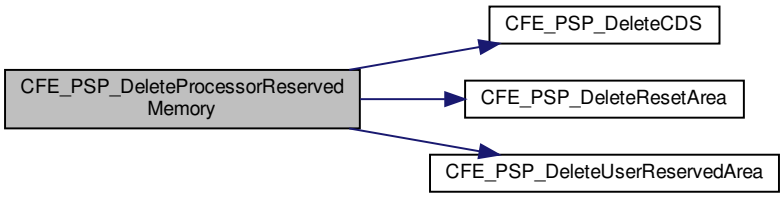
```
void CFE_PSP_DeleteProcessorReservedMemory (
             void  )
```

Definition at line 726 of file cfe_psp_memory.c.

References CFE_PSP_DeleteCDS(), CFE_PSP_DeleteResetArea(), and CFE_PSP_DeleteUserReservedArea().

Referenced by main().

Here is the call graph for this function:



13.103.2.3 CFE_PSP_DeleteResetArea()

```
void CFE_PSP_DeleteResetArea (
             void  )
```

Definition at line 397 of file cfe_psp_memory.c.

References ResetAreaShmId.

Referenced by CFE_PSP_DeleteProcessorReservedMemory().

13.103.2.4 CFE_PSP_DeleteUserReservedArea()

```
void CFE_PSP_DeleteUserReservedArea (
            void  )
```

Definition at line 527 of file cfe_psp_memory.c.

References UserShmId.

Referenced by CFE_PSP_DeleteProcessorReservedMemory().

13.103.2.5 CFE_PSP_GetCDSSize()

```
int32 CFE_PSP_GetCDSSize (
            uint32 * SizeOfCDS )
```

Definition at line 219 of file cfe_psp_memory.c.

References CFE_PSP_CDS_SIZE, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_CDS_EarlyInit().

13.103.2.6 CFE_PSP_GetCFETextSegmentInfo()

```
int32 CFE_PSP_GetCFETextSegmentInfo (
            cpuaddr * PtrToCFESegment,
            uint32 * SizeOfCFESegment )
```

Definition at line 781 of file cfe_psp_memory.c.

References _fini, _init, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_TaskInit().

13.103.2.7 CFE_PSP_GetKernelTextSegmentInfo()

```
int32 CFE_PSP_GetKernelTextSegmentInfo (
            cpuaddr * PtrToKernelSegment,
            uint32 * SizeOfKernelSegment )
```

Definition at line 753 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_ERROR_NOT_IMPLEMENTED, and NULL.

13.103.2.8 CFE_PSP_GetResetArea()

```
int32 CFE_PSP_GetResetArea (
            cpuaddr * PtrToResetArea,
            uint32 * SizeOfResetArea )
```

Definition at line 434 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_RESET_AREA_SIZE, CFE_PSP_ResetAreaPtr, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_ERLogDump(), CFE_ES_SetupResetVariables(), and CFE_EVS_EarlyInit().

13.103.2.9 CFE_PSP_GetUserReservedArea()

```
int32 CFE_PSP_GetUserReservedArea (
            cpuaddr * PtrToUserArea,
            uint32 * SizeOfUserArea )
```

Definition at line 559 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_SUCCESS, CFE_PSP_USER_RESERVED_SIZE, CFE_PSP_User↩ ReservedAreaPtr, and NULL.

13.103.2.10 CFE_PSP_GetVolatileDiskMem()

```
int32 CFE_PSP_GetVolatileDiskMem (
            cpuaddr * PtrToVolDisk,
            uint32 * SizeOfVolDisk )
```

Definition at line 624 of file cfe_psp_memory.c.

References CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_InitializeFileSystems().

13.103.2.11 CFE_PSP_InitCDS()

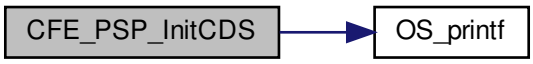
```
int32 CFE_PSP_InitCDS (
            uint32 RestartType )
```

Definition at line 128 of file cfe_psp_memory.c.

References CDSShmId, CFE_PSP_CDS_KEY_FILE, CFE_PSP_CDS_SIZE, CFE_PSP_CDSPtr, CFE_PSP_RST_↩ TYPE_POWERON, CFE_PSP_SUCCESS, and OS_printf().

Referenced by CFE_PSP_InitProcessorReservedMemory().

Here is the call graph for this function:



13.103.2.12 CFE_PSP_InitProcessorReservedMemory()

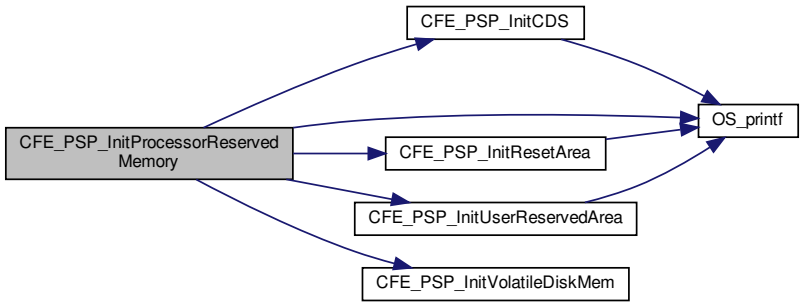
```
int32 CFE_PSP_InitProcessorReservedMemory (
            uint32 RestartType )
```

Definition at line 661 of file cfe_psp_memory.c.

References CFE_PSP_CDS_KEY_FILE, CFE_PSP_InitCDS(), CFE_PSP_InitResetArea(), CFE_PSP_InitUser↩ ReservedArea(), CFE_PSP_InitVolatileDiskMem(), CFE_PSP_RESERVED_KEY_FILE, CFE_PSP_RESET_KEY_F↩ ILE, CFE_PSP_SUCCESS, and OS_printf().

Referenced by main().

Here is the call graph for this function:



13.103.2.13 CFE_PSP_InitResetArea()

```
int32 CFE_PSP_InitResetArea (
            uint32 RestartType )
```

Definition at line 340 of file cfe_psp_memory.c.

References CFE_PSP_RESET_AREA_SIZE, CFE_PSP_RESET_KEY_FILE, CFE_PSP_ResetAreaPtr, CFE_PSP_↩ RST_TYPE_POWERON, CFE_PSP_SUCCESS, OS_printf(), and ResetAreaShmId.

Referenced by CFE_PSP_InitProcessorReservedMemory().

Here is the call graph for this function:

13.103.2.14 CFE_PSP_InitUserReservedArea()

```
int32 CFE_PSP_InitUserReservedArea (
            uint32 RestartType )
```

Definition at line 471 of file cfe_psp_memory.c.

References CFE_PSP_RESERVED_KEY_FILE, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_SUCCESS, CFE_↩ PSP_USER_RESERVED_SIZE, CFE_PSP_UserReservedAreaPtr, OS_printf(), and UserShmId.

Referenced by CFE_PSP_InitProcessorReservedMemory().

Here is the call graph for this function:

13.103.2.15 CFE_PSP_InitVolatileDiskMem()

```
int32 CFE_PSP_InitVolatileDiskMem (
            uint32 RestartType )
```

Definition at line 597 of file cfe_psp_memory.c.

References CFE_PSP_SUCCESS.

Referenced by CFE_PSP_InitProcessorReservedMemory().

13.103.2.16 CFE_PSP_ReadFromCDS()

```
int32 CFE_PSP_ReadFromCDS (
            void * PtrToDataToRead,
            uint32 CDSOffset,
            uint32 NumBytes )
```

Definition at line 292 of file cfe_psp_memory.c.

References CFE_PSP_CDS_SIZE, CFE_PSP_CDSPtr, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_CDSBlockRead(), CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_PutCD↩SBlock(), CFE_ES_RebuildCDS(), CFE_ES_RebuildCDSPool(), and CFE_ES_ValidateCDS().

13.103.2.17 CFE_PSP_WriteToCDS()

```
int32 CFE_PSP_WriteToCDS (
            const void * PtrToDataToWrite,
            uint32 CDSOffset,
            uint32 NumBytes )
```

Definition at line 249 of file cfe_psp_memory.c.

References CFE_PSP_CDS_SIZE, CFE_PSP_CDSPtr, CFE_PSP_ERROR, CFE_PSP_SUCCESS, and NULL.

Referenced by CFE_ES_CDSBlockWrite(), CFE_ES_GetCDSBlock(), CFE_ES_InitCDSRegistry(), CFE_ES_↩InitializeCDS(), CFE_ES_PutCDSBlock(), CFE_ES_RebuildCDSPool(), and CFE_ES_UpdateCDSRegistry().

13.103.3 Variable Documentation

13.103.3.1 _fini

```
unsigned int _fini
```

Referenced by CFE_PSP_GetCFETextSegmentInfo().

13.103.3.2 _init

```
unsigned int _init
```

Referenced by CFE_PSP_GetCFETextSegmentInfo().

13.103.3.3 CDSShmId

```
int CDSShmId
```

Definition at line 101 of file cfe_psp_memory.c.

Referenced by CFE_PSP_DeleteCDS(), and CFE_PSP_InitCDS().

13.103.3.4 CFE_PSP_CDSPtr

```
uint8* CFE_PSP_CDSPtr = 0
```

Definition at line 97 of file cfe_psp_memory.c.

Referenced by CFE_PSP_InitCDS(), CFE_PSP_ReadFromCDS(), and CFE_PSP_WriteToCDS().

13.103.3.5 CFE_PSP_ResetAreaPtr

```
uint8* CFE_PSP_ResetAreaPtr = 0
```

Definition at line 98 of file cfe_psp_memory.c.

Referenced by CFE_PSP_GetResetArea(), and CFE_PSP_InitResetArea().

13.103.3.6 CFE_PSP_UserReservedAreaPtr

```
uint8* CFE_PSP_UserReservedAreaPtr = 0
```

Definition at line 99 of file cfe_psp_memory.c.

Referenced by CFE_PSP_GetUserReservedArea(), and CFE_PSP_InitUserReservedArea().

13.103.3.7 ResetAreaShmId

```
int ResetAreaShmId
```

Definition at line 100 of file cfe_psp_memory.c.

Referenced by CFE_PSP_DeleteResetArea(), and CFE_PSP_InitResetArea().

13.103.3.8 UserShmId

```
int UserShmId
```

Definition at line 102 of file cfe_psp_memory.c.

Referenced by CFE_PSP_DeleteUserReservedArea(), and CFE_PSP_InitUserReservedArea().

13.104 psp/fsw/pc-linux/src/cfe_psp_memtab.c File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
#include "cfe_psp_config.h"
```

Variables

- CFE_PSP_MemTable_t CFE_PSP_MemoryTable [CFE_PSP_MEM_TABLE_SIZE]

13.104.1 Variable Documentation

13.104.1.1 CFE_PSP_MemoryTable

CFE_PSP_MemTable_t CFE_PSP_MemoryTable[CFE_PSP_MEM_TABLE_SIZE]

Initial value:

```
=
{
   { CFE_PSP_MEM_RAM, CFE_PSP_MEM_SIZE_DWORD, 0, 0xFFFFFFFF,
     CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
   { CFE_PSP_MEM_INVALID, 0, 0, 0, CFE_PSP_MEM_ATTR_READWRITE },
}
```

Extern reference to the psp memory table Allows the actual instantiation to be done outside this module

Definition at line 46 of file cfe_psp_memtab.c.

13.105 psp/fsw/pc-linux/src/cfe_psp_ssr.c File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
#include "cfe_psp_config.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

Functions

- int32 CFE_PSP_InitSSR (uint32 bus, uint32 device, char *DeviceName)

13.105.1 Function Documentation

13.105.1.1 CFE_PSP_InitSSR()

```
int32 CFE_PSP_InitSSR (
            uint32 bus,
            uint32 device,
            char * DeviceName )
```

Definition at line 66 of file cfe_psp_ssr.c.

References CFE_PSP_ERROR.

13.106 psp/fsw/pc-linux/src/cfe_psp_start.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <sys/time.h>
#include <getopt.h>
#include <limits.h>
#include <pthread.h>
#include <sched.h>
#include <errno.h>
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
#include <target_config.h>
#include "cfe_psp_module.h"
```

Data Structures

- struct CFE_PSP_CommandData_t

Macros

- #define CFE_PSP_MAIN_FUNCTION (∗GLOBAL_CONFIGDATA.CfeConfig->SystemMain)
- #define CFE_PSP_1HZ_FUNCTION (∗GLOBAL_CONFIGDATA.CfeConfig->System1HzISR)
- #define CFE_PSP_NONVOL_STARTUP_FILE (GLOBAL_CONFIGDATA.CfeConfig->NonvolStartupFile)
- #define CFE_PSP_CPU_ID (GLOBAL_CONFIGDATA.Default_CpuId)
- #define CFE_PSP_CPU_NAME (GLOBAL_CONFIGDATA.Default_CpuName)
- #define CFE_PSP_SPACECRAFT_ID (GLOBAL_CONFIGDATA.Default_SpacecraftId)
- #define CFE_PSP_CPU_NAME_LENGTH 32
- #define CFE_PSP_RESET_NAME_LENGTH 10

Functions

- void CFE_PSP_SigintHandler (int signal)
- void CFE_PSP_TimerHandler (int signum)
- void CFE_PSP_DisplayUsage (char ∗Name)
- void CFE_PSP_ProcessArgumentDefaults (CFE_PSP_CommandData_t ∗CommandDataDefault)
- void CFE_PSP_SetupLocal1Hz (void)
- void CFE_PSP_DeleteProcessorReservedMemory (void)
- int main (int argc, char ∗argv[ ])

Variables

- uint32 TimerCounter
- CFE_PSP_CommandData_t CommandData
- uint32 CFE_PSP_SpacecraftId
- uint32 CFE_PSP_CpuId
- char CFE_PSP_CpuName [CFE_PSP_CPU_NAME_LENGTH]
- static const char * optString = "R:S:C:I:N:h"
- static const struct option longOpts [ ]

13.106.1 Macro Definition Documentation

13.106.1.1 CFE_PSP_1HZ_FUNCTION

```
#define CFE_PSP_1HZ_FUNCTION (*GLOBAL_CONFIGDATA.CfeConfig->System1HzISR)
```

Definition at line 67 of file cfe_psp_start.c.

Referenced by CFE_PSP_TimerHandler().

13.106.1.2 CFE_PSP_CPU_ID

```
#define CFE_PSP_CPU_ID (GLOBAL_CONFIGDATA.Default_CpuId)
```

Definition at line 69 of file cfe_psp_start.c.

Referenced by CFE_PSP_DisplayUsage(), and CFE_PSP_ProcessArgumentDefaults().

13.106.1.3 CFE_PSP_CPU_NAME

```
#define CFE_PSP_CPU_NAME (GLOBAL_CONFIGDATA.Default_CpuName)
```

Definition at line 70 of file cfe_psp_start.c.

Referenced by CFE_PSP_DisplayUsage(), and CFE_PSP_ProcessArgumentDefaults().

13.106.1.4 CFE_PSP_CPU_NAME_LENGTH

```
#define CFE_PSP_CPU_NAME_LENGTH 32
```

Definition at line 77 of file cfe_psp_start.c.

Referenced by CFE_PSP_ProcessArgumentDefaults(), and main().

13.106.1.5 CFE_PSP_MAIN_FUNCTION

```
#define CFE_PSP_MAIN_FUNCTION (*GLOBAL_CONFIGDATA.CfeConfig->SystemMain)
```

Definition at line 66 of file cfe_psp_start.c.

Referenced by main().

13.106.1.6 CFE_PSP_NONVOL_STARTUP_FILE

```
#define CFE_PSP_NONVOL_STARTUP_FILE (GLOBAL_CONFIGDATA.CfeConfig->NonvolStartupFile)
```

Definition at line 68 of file cfe_psp_start.c.

Referenced by main().

13.106.1.7 CFE_PSP_RESET_NAME_LENGTH

```
#define CFE_PSP_RESET_NAME_LENGTH 10
```

Definition at line 78 of file cfe_psp_start.c.

Referenced by main().

13.106.1.8 CFE_PSP_SPACECRAFT_ID

```
#define CFE_PSP_SPACECRAFT_ID (GLOBAL_CONFIGDATA.Default_SpacecraftId)
```

Definition at line 71 of file cfe_psp_start.c.

Referenced by CFE_PSP_DisplayUsage(), and CFE_PSP_ProcessArgumentDefaults().

13.106.2 Function Documentation

13.106.2.1 CFE_PSP_DeleteProcessorReservedMemory()

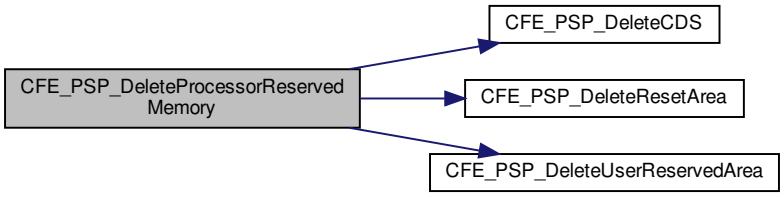
```
void CFE_PSP_DeleteProcessorReservedMemory (
            void  )
```

Definition at line 726 of file cfe_psp_memory.c.

References CFE_PSP_DeleteCDS(), CFE_PSP_DeleteResetArea(), and CFE_PSP_DeleteUserReservedArea().

Referenced by main().

Here is the call graph for this function:



13.106.2.2 CFE_PSP_DisplayUsage()

```
void CFE_PSP_DisplayUsage (
            char * Name )
```

Definition at line 439 of file cfe_psp_start.c.

References CFE_PSP_CPU_ID, CFE_PSP_CPU_NAME, and CFE_PSP_SPACECRAFT_ID.

Referenced by main().

13.106.2.3 CFE_PSP_ProcessArgumentDefaults()

```
void CFE_PSP_ProcessArgumentDefaults (
            CFE_PSP_CommandData_t * CommandDataDefault )
```

Definition at line 487 of file cfe_psp_start.c.

References CFE_PSP_CPU_ID, CFE_PSP_CPU_NAME, CFE_PSP_CPU_NAME_LENGTH, CFE_PSP_SPACEC↩RAFT_ID, CFE_PSP_CommandData_t::CpuId, CFE_PSP_CommandData_t::CpuName, CFE_PSP_CommandData↩_t::GotCpuId, CFE_PSP_CommandData_t::GotCpuName, CFE_PSP_CommandData_t::GotResetType, CFE_PSP_↩CommandData_t::GotSpacecraftId, CFE_PSP_CommandData_t::GotSubType, CFE_PSP_CommandData_t::Reset↩Type, CFE_PSP_CommandData_t::SpacecraftId, and CFE_PSP_CommandData_t::SubType.

Referenced by main().

13.106.2.4 CFE_PSP_SetupLocal1Hz()

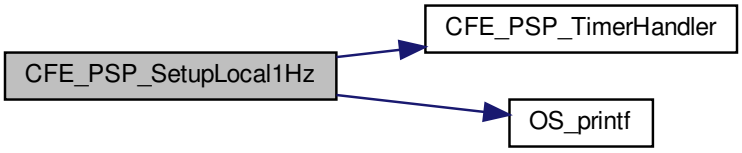
```
void CFE_PSP_SetupLocal1Hz (
             void  )
```

Definition at line 550 of file cfe_psp_start.c.

References CFE_PSP_TimerHandler(), NULL, OS_printf(), and TimerCounter.

Referenced by main().

Here is the call graph for this function:



13.106.2.5 CFE_PSP_SigintHandler()

```
void CFE_PSP_SigintHandler (
             int signal )
```

Definition at line 398 of file cfe_psp_start.c.

References OS_ApplicationShutdown().

Referenced by main().

Here is the call graph for this function:

13.106.2.6 CFE_PSP_TimerHandler()

```
void CFE_PSP_TimerHandler (
            int signum )
```

Definition at line 416 of file cfe_psp_start.c.

References CFE_PSP_1HZ_FUNCTION, and TimerCounter.

Referenced by CFE_PSP_SetupLocal1Hz().

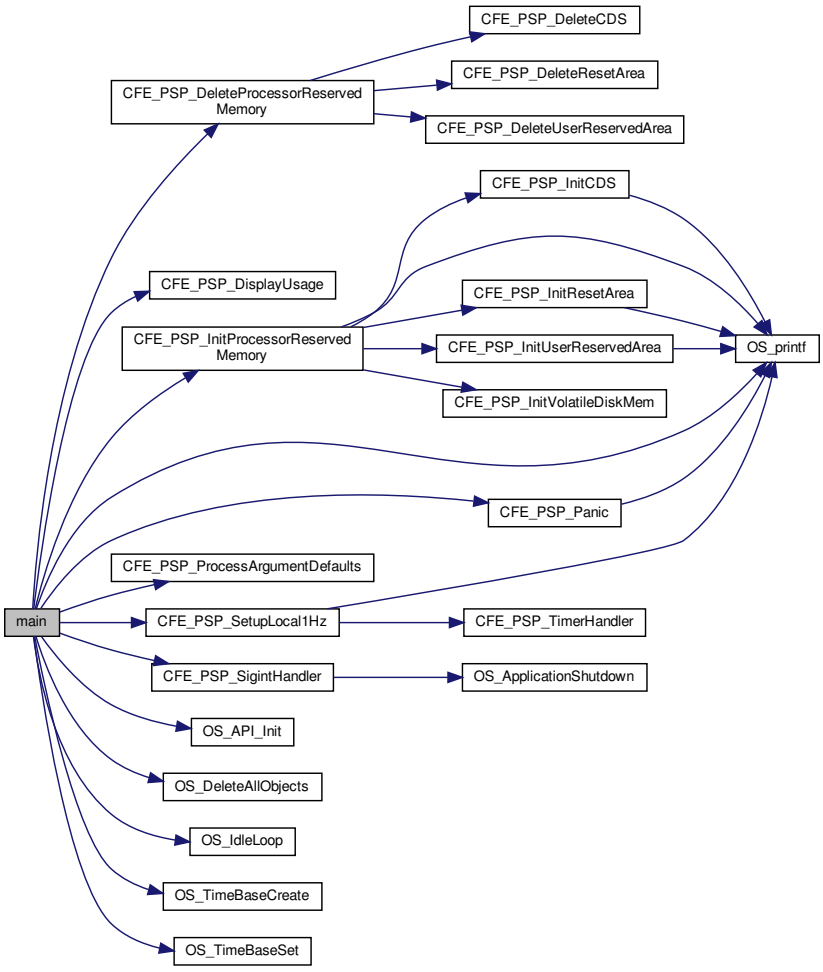
13.106.2.7 main()

```
int main (
            int argc,
            char * argv[] )
```

Definition at line 160 of file cfe_psp_start.c.

References CFE_PSP_CPU_NAME_LENGTH, CFE_PSP_CpuId, CFE_PSP_CpuName, CFE_PSP_Delete↩ ProcessorReservedMemory(), CFE_PSP_DisplayUsage(), CFE_PSP_InitProcessorReservedMemory(), CFE_PS↩ P_MAIN_FUNCTION, CFE_PSP_NONVOL_STARTUP_FILE, CFE_PSP_Panic(), CFE_PSP_ProcessArgument↩ Defaults(), CFE_PSP_RESET_NAME_LENGTH, CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PR↩ OCESSOR, CFE_PSP_SetupLocal1Hz(), CFE_PSP_SigintHandler(), CFE_PSP_SpacecraftId, CFE_PSP_SUCCESS, CFE_PSP_CommandData_t::CpuId, CFE_PSP_CommandData_t::CpuName, CFE_PSP_CommandData_t::GotCpu↩ Id, CFE_PSP_CommandData_t::GotCpuName, CFE_PSP_CommandData_t::GotResetType, CFE_PSP_Command↩ Data_t::GotSpacecraftId, CFE_PSP_CommandData_t::GotSubType, longOpts, NULL, optString, OS_API_Init(), OS↩ _DeleteAllObjects(), OS_IdleLoop(), OS_printf(), OS_SUCCESS, OS_TimeBaseCreate(), OS_TimeBaseSet(), CFE↩ _PSP_CommandData_t::ResetType, CFE_PSP_CommandData_t::SpacecraftId, and CFE_PSP_CommandData_t::↩ SubType.

Here is the call graph for this function:



13.106.3 Variable Documentation

13.106.3.1 CFE_PSP_CpuId

```
uint32 CFE_PSP_CpuId
```

Definition at line 126 of file cfe_psp_start.c.

Referenced by CFE_PSP_GetProcessorId(), and main().

13.106.3.2 CFE_PSP_CpuName

```
char CFE_PSP_CpuName[CFE_PSP_CPU_NAME_LENGTH]
```

Definition at line 127 of file cfe_psp_start.c.

Referenced by main().

13.106.3.3 CFE_PSP_SpacecraftId

```
uint32 CFE_PSP_SpacecraftId
```

Definition at line 125 of file cfe_psp_start.c.

Referenced by CFE_PSP_GetSpacecraftId(), and main().

13.106.3.4 CommandData

```
CFE_PSP_CommandData_t CommandData
```

Definition at line 124 of file cfe_psp_start.c.

13.106.3.5 longOpts

```
const struct option longOpts[]  [static]
```

Initial value:

```
= {
   { "reset",     required_argument, NULL, 'R' },
   { "subtype",   required_argument, NULL, 'S' },
   { "cpuid",     required_argument, NULL, 'C' },
   { "scid",      required_argument, NULL, 'I'},
   { "cpuname",   required_argument, NULL, 'N'},
   { "help",      no_argument,       NULL, 'h' },
   { NULL,        no_argument,       NULL,  0 }
}
```

Definition at line 137 of file cfe_psp_start.c.

Referenced by main().

13.106.3.6 optString

```
const char* optString = "R:S:C:I:N:h"  [static]
```

Definition at line 132 of file cfe_psp_start.c.

Referenced by main().

13.106.3.7 TimerCounter

```
uint32 TimerCounter
```

Definition at line 123 of file cfe_psp_start.c.

Referenced by CFE_PSP_SetupLocal1Hz(), and CFE_PSP_TimerHandler().

13.107 psp/fsw/pc-linux/src/cfe_psp_support.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp.h"
```

Functions

- void CFE_PSP_Restart (uint32 reset_type)
- void CFE_PSP_Panic (int32 ErrorCode)
- void CFE_PSP_FlushCaches (uint32 type, cpuaddr address, uint32 size)
- uint32 CFE_PSP_GetProcessorId (void)
- uint32 CFE_PSP_GetSpacecraftId (void)

Variables

- uint32 CFE_PSP_SpacecraftId
- uint32 CFE_PSP_CpuId

13.107.1 Function Documentation

13.107.1.1 CFE_PSP_FlushCaches()

```
void CFE_PSP_FlushCaches (
              uint32 type,
              cpuaddr address,
              uint32 size )
```

Definition at line 125 of file cfe_psp_support.c.

13.107.1.2 CFE_PSP_GetProcessorId()

```
uint32 CFE_PSP_GetProcessorId (
              void  )
```

Definition at line 147 of file cfe_psp_support.c.

References CFE_PSP_CpuId.

Referenced by CFE_SB_SendMsgFull(), and EVS_GenerateEventTelemetry().

13.107.1.3 CFE_PSP_GetSpacecraftId()

```
uint32 CFE_PSP_GetSpacecraftId (
              void  )
```

Definition at line 168 of file cfe_psp_support.c.

References CFE_PSP_SpacecraftId.

Referenced by EVS_GenerateEventTelemetry().

13.107.1.4 CFE_PSP_Panic()

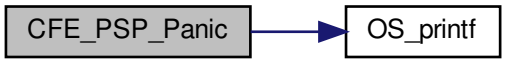
```
void CFE_PSP_Panic (
              int32 ErrorCode )
```

Definition at line 104 of file cfe_psp_support.c.

References OS_printf().

Referenced by CFE_ES_CreateObjects(), CFE_ES_InitializeFileSystems(), CFE_ES_Main(), CFE_ES_SetupReset↩ Variables(), and main().

Here is the call graph for this function:



13.107.1.5 CFE_PSP_Restart()

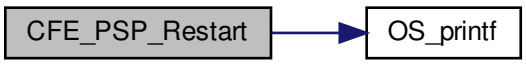
```
void CFE_PSP_Restart (
             uint32 reset_type )
```

Definition at line 70 of file cfe_psp_support.c.

References CFE_PSP_RST_TYPE_POWERON, CFE_PSP_RST_TYPE_PROCESSOR, and OS_printf().

Referenced by CFE_ES_ProcessCoreException(), CFE_ES_ResetCFE(), and CFE_ES_SetupResetVariables().

Here is the call graph for this function:



13.107.2 Variable Documentation

13.107.2.1 CFE_PSP_CpuId

```
uint32 CFE_PSP_CpuId
```

Definition at line 126 of file cfe_psp_start.c.

Referenced by CFE_PSP_GetProcessorId(), and main().

13.107.2.2 CFE_PSP_SpacecraftId

```
uint32 CFE_PSP_SpacecraftId
```

Definition at line 125 of file cfe_psp_start.c.

Referenced by CFE_PSP_GetSpacecraftId(), and main().

13.108 psp/fsw/pc-linux/src/cfe_psp_timer.c File Reference

```
#include "common_types.h"
#include "osapi.h"
#include <stdio.h>
#include <stdlib.h>
#include "cfe_psp.h"
```

Macros

- #define CFE_PSP_TIMER_TICKS_PER_SECOND
- #define CFE_PSP_TIMER_LOW32_ROLLOVER

Functions

- void CFE_PSP_GetTime (OS_time_t *LocalTime)
- uint32 CFE_PSP_Get_Timer_Tick (void)
- uint32 CFE_PSP_GetTimerTicksPerSecond (void)
- uint32 CFE_PSP_GetTimerLow32Rollover (void)
- void CFE_PSP_Get_Timebase (uint32 *Tbu, uint32 *Tbl)
- uint32 CFE_PSP_Get_Dec (void)

13.108.1 Macro Definition Documentation

13.108.1.1 CFE_PSP_TIMER_LOW32_ROLLOVER

```
#define CFE_PSP_TIMER_LOW32_ROLLOVER
```

Value:

```
1000000     /* The number that the least significant 32 bits of the 64 bit
                                                    time stamp returned by OS_BSPGet_Timebase rolls
       over.  If the lower 32
                                                    bits rolls at 1 second, then the
       OS_BSP_TIMER_LOW32_ROLLOVER will be 1000000.
                                                    if the lower 32 bits rolls at its maximum value
       (2^32) then
                                                    OS_BSP_TIMER_LOW32_ROLLOVER will be 0. */
```

Definition at line 63 of file cfe_psp_timer.c.

Referenced by CFE_PSP_GetTimerLow32Rollover().

13.108.1.2 CFE_PSP_TIMER_TICKS_PER_SECOND

```
#define CFE_PSP_TIMER_TICKS_PER_SECOND
```

Value:

```
1000000     /* Resolution of the least significant 32 bits of the 64 bit
                                                        time stamp returned by OS_BSPGet_Timebase in
       timer ticks per second.
                                                        The timer resolution for accuracy should not be
       any slower than 1000000
                                                        ticks per second or 1 us per tick */
```

Definition at line 59 of file cfe_psp_timer.c.

Referenced by CFE_PSP_GetTimerTicksPerSecond().

13.108.2 Function Documentation

13.108.2.1 CFE_PSP_Get_Dec()

```
uint32 CFE_PSP_Get_Dec (
            void  )
```

Definition at line 185 of file cfe_psp_timer.c.

13.108.2.2 CFE_PSP_Get_Timebase()

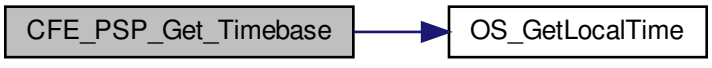
```
void CFE_PSP_Get_Timebase (
            uint32 * Tbu,
            uint32 * Tbl )
```

Definition at line 162 of file cfe_psp_timer.c.

References OS_time_t::microsecs, OS_GetLocalTime(), and OS_time_t::seconds.

Referenced by CFE_ES_PerfLogAdd().

Here is the call graph for this function:



13.108.2.3 CFE_PSP_Get_Timer_Tick()

```
uint32 CFE_PSP_Get_Timer_Tick (
            void  )
```

Definition at line 102 of file cfe_psp_timer.c.

13.108.2.4 CFE_PSP_GetTime()

```
void CFE_PSP_GetTime (
            OS_time_t * LocalTime )
```

Definition at line 77 of file cfe_psp_timer.c.

References OS_GetLocalTime().

Here is the call graph for this function:

13.108.2.5 CFE_PSP_GetTimerLow32Rollover()

```
uint32 CFE_PSP_GetTimerLow32Rollover (
            void  )
```

Definition at line 144 of file cfe_psp_timer.c.

References CFE_PSP_TIMER_LOW32_ROLLOVER.

Referenced by CFE_ES_SetupPerfVariables().

13.108.2.6 CFE_PSP_GetTimerTicksPerSecond()

```
uint32 CFE_PSP_GetTimerTicksPerSecond (
            void  )
```

Definition at line 123 of file cfe_psp_timer.c.

References CFE_PSP_TIMER_TICKS_PER_SECOND.

Referenced by CFE_ES_SetupPerfVariables().

13.109 psp/fsw/pc-linux/src/cfe_psp_voltab.c File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "osconfig.h"
```

Variables

- OS_VolumeInfo_t OS_VolumeTable [NUM_TABLE_ENTRIES]

13.109.1 Variable Documentation

13.109.1.1 OS_VolumeTable

```
OS_VolumeInfo_t OS_VolumeTable[NUM_TABLE_ENTRIES]
```

Initial value:

```
=
{
{"/ramdev0", "./ram",      FS_BASED,       true,      true,     false,     " ",       " ",      0
            },
{"/ramdev1", "./ram1",     FS_BASED,       true,      true,     false,     " ",       " ",      0
            },
{"/ramdev2", "./ram2",     FS_BASED,       true,      true,     false,     " ",       " ",      0
            },
{"/ramdev3", "./ram3",     FS_BASED,       true,      true,     false,     " ",       " ",      0
            },
{"/ramdev4", "./ram4",     FS_BASED,       true,      true,     false,     " ",       " ",      0
            },


{"/eedev0",  "./cf",      FS_BASED,       false,     false,    true,     "CF",      "/cf",     512
         },

{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          },
{"unused",   "unused",    FS_BASED,       true,      true,     false,     " ",       " ",      0
          }
}
```

Extern reference to the psp volume table Allows the actual instantiation to be done outside this module

Definition at line 63 of file cfe_psp_voltab.c.

13.110 psp/fsw/pc-linux/src/cfe_psp_watchdog.c File Reference

```
#include "common_types.h"
#include "osapi.h"
#include <stdio.h>
#include <stdlib.h>
#include "cfe_psp.h"
#include "cfe_psp_config.h"
```

Functions

- void CFE_PSP_WatchdogInit (void)
- void CFE_PSP_WatchdogEnable (void)
- void CFE_PSP_WatchdogDisable (void)
- void CFE_PSP_WatchdogService (void)
- uint32 CFE_PSP_WatchdogGet (void)
- void CFE_PSP_WatchdogSet (uint32 WatchdogValue)

Variables

- uint32 CFE_PSP_WatchdogValue = CFE_PSP_WATCHDOG_MAX

13.110.1 Function Documentation

13.110.1.1 CFE_PSP_WatchdogDisable()

```
void CFE_PSP_WatchdogDisable (
            void  )
```

Definition at line 114 of file cfe_psp_watchdog.c.

13.110.1.2 CFE_PSP_WatchdogEnable()

```
void CFE_PSP_WatchdogEnable (
            void  )
```

Definition at line 98 of file cfe_psp_watchdog.c.

13.110.1.3 CFE_PSP_WatchdogGet()

```
uint32 CFE_PSP_WatchdogGet (
            void  )
```

Definition at line 156 of file cfe_psp_watchdog.c.

References CFE_PSP_WatchdogValue.

13.110.1.4 CFE_PSP_WatchdogInit()

```
void CFE_PSP_WatchdogInit (
            void  )
```

Definition at line 75 of file cfe_psp_watchdog.c.

References CFE_PSP_WatchdogValue.

13.110.1.5 CFE_PSP_WatchdogService()

```
void CFE_PSP_WatchdogService (
            void  )
```

Definition at line 135 of file cfe_psp_watchdog.c.

13.110.1.6 CFE_PSP_WatchdogSet()

```
void CFE_PSP_WatchdogSet (
            uint32 WatchdogValue )
```

Definition at line 177 of file cfe_psp_watchdog.c.

References CFE_PSP_WatchdogValue.

13.110.2 Variable Documentation

13.110.2.1 CFE_PSP_WatchdogValue

```
uint32 CFE_PSP_WatchdogValue = CFE_PSP_WATCHDOG_MAX
```

Definition at line 64 of file cfe_psp_watchdog.c.

Referenced by CFE_PSP_WatchdogGet(), CFE_PSP_WatchdogInit(), and CFE_PSP_WatchdogSet().